涉密论文 □　公开论文 □

浙 江 大 學

本科生毕业论文

题目 基于对流扩散方程关于物理信息机器学习算法的研究

姓名与学号　陈冠宇 3200102033

指导教师　徐翔，倪东

年级与专业　2020级数学与应用数学 (强基计划)

所在学院　数学科学学院

递交日期　2024 年 5 月 31 号

浙江大学本科生毕业论文（设计）承诺书

1. 本人郑重地承诺所呈交的毕业论文（设计），是在指导教师的指导下严格按照学校和学院有关规定完成的。

2. 本人在毕业论文（设计）中除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得**浙江大学**或其他教育机构的学位或证书而使用过的材料。

3. 与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

4. 本人承诺在毕业论文（设计）工作过程中没有伪造数据等行为。

5. 若在本毕业论文（设计）中有侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

6. 本人完全了解**浙江大学**有权保留并向有关部门或机构送交本论文（设计）的复印件和磁盘，允许本论文（设计）被查阅和借阅。本人授权**浙江大学**可以将本论文（设计）的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编本论文（设计）。

作者签名：　　导师签名：

签字日期：　　年　月　日　　签字日期　　年　月　日

致谢

在求是园里的四年时光飞逝，虽有两年疫情，略显遗憾，但总体来说还算充实。首先我的导师倪东老师，在完成论文的过程中给予了很多的指导意见。感谢本科四年间的所有老师们。在数学学院学习的日子里，他们让我收获很多，让我学到了非常多专业且实用的知识。其次感谢身边的同学和朋友，强基数学这个家庭，虽然不大，但仍让我感觉到这是一个温暖的整体，让这四年的每一天都有记忆。感谢我的室友，吕梓沐，王锦宸和黄文翀。还要感谢学长学姐，无论是学业还是生活上的帮助，感谢徐圣泽，邵国江，郭其龙，顾皓，张洪申等学长的热情帮助和分享，他们真的帮助了我很多。最后当然还要感谢我的家人，在这一路上的支持，无论是经济上还是精神上，感谢他们对我一切选择的支持，我爱你们。

紫金港很美，浙里很美，杭州很美。我爱浙大，爱这里的人，感觉非常幸福。

摘要

这项工作的主要目标是完成对经典数值算法和深度学习算法的结合，具体来说是间断有限元方法和神经网络的结合。在文章中我们选取了 Possion 方程和热方程作为方法研究的基本模型，针对提出的方法论考察了其理论和数值方面的可行性。

文章分为四个部分，分别是经典数值方法的理论和数值实验结果以及实施细节，和深度学习算法相结合的理论设计和数值实验的结果和实施细节。具体来讲内容如下：

第一部分首先对于一些经典的数值方法做一些调研工作，尤其是有限元类方法，包括混合有限元方法以及间断有限元方法。其次由于求解目标并不是一个稳态方程，所以我们讨论了关于解决瞬态 PDE 问题相关的经典数值方法，如线性多步法。但是在这里我们还是主要介绍一种不同于一般处理时间项的数值方法，时空混合方法（Space-Time Method），并且将其和间断有限元方法结合起来，并且基于一维热方程和 NS 方程介绍其理论。

第二部分首先介绍了数值实现的实施细节，包括理论设计和代码设计，并且主要基于上述经典数值方法完成了数值实验，1. 关于 Possion 方程，主要讨论了规则区域下齐次边界条件下和非齐次边界条件下的结果以及不规则边界下的效果, 以及有奇点的情况；2.Heat 方程，主要讨论了有规则解和不规则解的情况，完成相关数值实验。

第三部分首先介绍了目前主流的求解 PDE 系统的相关深度学习方法，其中着重介绍了和 PDE 弱解形式相关的两种方法：弱对抗神经网络以及 ParticleWNN。随后根据间断有限元给出的启发，基于现有方法提出改进方案。这里主要介绍想法以及方案设计思路。

第四部分基于一维和二维的 Possion 方程展示提出方案的数值可行性，展示实验结果，并对结果进行总结。

最后对于整篇工作进行总结，并提出研究展望。

关键词： 间断伽辽金方法；时空方法；物理信息神经网络；弱解形式；基函数；

Abstract

In this work, our main goal is to combine classical numerical methods with deep learning techniques. Focusing on the Poisson equation and the heat equation, we present both the theory and the results of related numerical experiments.

Our work is divided into four parts: the theory and numerical experiments of the (Space-Time) discontinuous Galerkin method, and the theoretical design and numerical experiments based on a deep learning framework. Specifically, we have:

In the first part, we commence with research on classical numerical methods, particularly finite element methods, which include mixed finite element methods and Discontinuous Galerkin methods. Additionally, we explore classical numerical methods for solving time-dependent problems, such as linear multi-step methods. However, here we primarily introduce a novel numerical method, the Space Time Method (STM). Building upon STM, we integrate it with the discontinuous finite element method and present its theory based on the one-dimensional heat equation and the Navier-Stokes equation.

In the second part, we begin by presenting the details of numerical implementation, encompassing both theoretical and code design aspects. Subsequently, we conduct numerical experiments based on the classical numerical methods mentioned earlier. Specifically:

For the Poisson equation, we discuss the results under homogeneous and nonhomogeneous boundary conditions in regular and irregular domains, and also solution with singular point. For the heat equation, we mainly explore cases of regular and irregular solutions, completing relevant numerical experiments using STM.

In the third part, we introduce current deep learning frameworks for solving PDE systems, focusing primarily on methods related to weak solutions of PDEs, such as weak adversarial networks and ParticleWNN. Inspired by the discontinuous finite element method, we propose a new approach, (Split) DGNet, based on existing methods, primarily discussing ideas and frameworks of design.

In the last part, we demonstrate the practicality of the new scheme and present numerical results based on the Poisson and heat equations.

Finally, we conclude by summarizing our work and presenting prospects for future research.

Keywords: Discontinuous Galerkin Method; Space-Time Method; Physics-informed Nerual Network; Weak Solution; Basis functions;

目录

第一部分　毕业论文

第一部分

毕业论文

一、绪论

1 研究背景

近些年随着计算机算力以及数值算法的不断发展，偏微分方程数值解方面有了很大的进步。对于偏微分方程而言，它在科学、工程、经济和金融领域有着广泛的应用。因此求解此类问题的方法研究具有重大的理论和实际工程价值。然而对于偏微分方程而言，仅有少部分方程能够由分析的方法得到其精确的解析结果，对于绝大多数的偏微分方程而言，由于其一些高度非线性项，其解析解几乎不能显式表达出来。因此如何精准的模拟并预测这些场的变化，如流场，电磁场甚至是多物理场，是现在比较棘手的问题。

为此我们发展了 PDE 经典的数值解法，例如有限差分方法 (FDM)，有限体积方法 (FVM)，有限元方法 (FEM) 等。但是随着对精度，模拟速度要求的提高，传统计算方法也显现出一些弊端。而随着深度学习算法的进步，深度学习工具开始为解决这些问题注入新的活力。

2 研究现状

在经典数值算法方面，目前对于间断伽辽金方法并没有具体的简易实现代码。并且对于瞬态 PDE 往往以一种时间差分离散进行时间步长的迭代算法，这样很容易出现误差累积的问题，并且对于动边界的问题很难处理。

深度学习算法方面，虽然已经有利用强形式以及数据进行学习的深度学习范式，但是针对弱解形式方程的处理方案少之又少。然而弱解在求解 PDE 中至关重要，尤其是对于有间断解或是跳跃的边界条件问题上。因此对于此类问题迫切需要一些新的方法。

3 研究目的

主要实现间断伽辽金的数值算法，并且将 DG 方法应用到时空方案中，这样对于时间维度的间断问题上有很大优势。

在此基础上，结合深度学习的优势，提出一种融合 (ST)DG 方法的深度学习框架，以达到求解加速，提高精度的效果。

二、理论部分

1 一些记号

首先定义一些记号。Ω 为数值解区域；$\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$: 为混合边界条件，例如 Dirichlet 边界以及 Neumann 边界条件。$\mathscr{T}_h$: 区域 Ω 三角化之后离散区域，h 为衡量标准，例如三角单元最长边。

$\mathscr{E}_h$: 离散化区域 $\mathscr{T}_h$ 中所有边的集合；具体的 $\mathscr{E}_h^I$: 离散化区域 $\mathscr{T}_h$ 中内部边的集合, $\mathscr{E}_h^D$: $\mathscr{T}_h \cap \partial\Omega_D$ 上边的集合，$\mathscr{E}_h^N$: $\mathscr{T}_h \cap \partial\Omega_N$ 中边的集合。$\mathscr{V}_h$: 离散化区域 $\mathscr{T}_h$ 中所有点的集合。且记 τ 为离散化区域 $\mathscr{T}_h$ 中一个单元.

$\Gamma = \partial\Omega, N_{\mathscr{E}} = card(\mathscr{E}_h), N_v = card(\mathscr{V}_h), N_k = card(\mathscr{T}_h), (\cdot,\cdot)$ 和 $<\cdot,\cdot>$ 分别是 Ω, Γ 上的内积记号。

由于我们考虑二维上的问题，因此有必要介绍一些关于向量导数的记号。

粗体字母记号，例如 $\mathbf{u}$ 代表 u 为一个向量, $(u_1, u_2, \cdots u_n)$, 其中 n 为定义域维度。对于 2D, 则有 $\mathbf{u} = (u_1, u_2)$.

那么

$$\nabla\mathbf{u} = \begin{pmatrix} u_{1x} & u_{2x} \\ u_{1y} & u_{2y} \end{pmatrix}, \nabla\cdot(\nabla\mathbf{u}) = \Delta\mathbf{u} = \begin{pmatrix} u_{1xx} + u_{1yy} \\ u_{2xx} + u_{2yy} \end{pmatrix}$$

为了计算弱形式的方便，我们引入 double product:

$$\nabla\mathbf{u} : \nabla\mathbf{v} = \sum_j \sum_i \frac{\partial u_i}{\partial x_j}\frac{\partial v_i}{\partial x_j}$$

和索伯列夫空间 (Sobolev Space)

$$H^1(\Omega) = \{u \in L^2(\Omega) : \partial^\alpha u \in L^2(\Omega), |\alpha| \le 1\}$$

$$H_0^1(\Omega) = \{u \in H^1(\Omega) : u|_{\partial\Omega} = 0\}$$

$$L_0^2(\Omega) = \{u \in L^2(\Omega) : \int_\Omega u = 0\}$$

对于向量 $\mathbf{u} \in [H_0^1]^n$, 有范数

$$\|\mathbf{u}\|_{H_0^1} = \left(\|\mathbf{u}\|_{L^2(\Omega)}^2 + \|\nabla\cdot\mathbf{u}\|_{L^2(\Omega)}^2\right)^{\frac{1}{2}}$$

接下来研究经典数值算法中的间断 Galerkin 方法。

2 间断有限元方法 (Discontinuous Galerkin Method)

间断有限元方法[1]是属于有限元类的数值方法，大致思路和一般有限元方法类似，选取基函数，最后求解为基函数的线性表示。最先被设计用于解决双曲守恒定律的经典数值算法，在近些年里逐渐变得流行。其主要特点在于允许解有间断性，这在解决拥有间断，突变解的问题上有着很大的优势。

我们选择间断有限元方法主要是由于它以下灵活性：

- 允许利用任意已知点，在任意区域内进行三角划分
- 在每个单元上可以自由的选择用于拟合的多项式的次数
- 可以实现很高效率的并行计算

间断有限元方法也有很多方式，如内罚间断，局部间断以及杂交间断等方式[2]。在这里我们主要研究内罚间断有限元方法 (Interior Penalty Discontinuous Galerkin Method)，并以解决 Possion 方程为例。

首先因为 DG 方法本身允许解的间断，有如下定义：

Definition 1 (Jump and average)**.** 定义边上跳量和平均量：

$$\begin{cases} [u]_{kl} = (u|_{\tau_k} - u|_{\tau_l}) \\ u = \dfrac{1}{2}(u|_{\tau_k} + u|_{\tau_l}) \end{cases}$$

其中 *kl* 为边 $e \in \mathscr{E}^I$ 邻居单元 *k* 和单元 *l*。

若 $e \in \mathscr{E}^D$, 只有邻居单元 *k*, 则定义为

$$\begin{cases} [u]_k = u|_{\tau_k} \\ u = u|_{\tau_k} \end{cases}$$

Definition 2 (Broken Sobolev space)**.** 对于 $p \geq 0$ *Broken Sobolev space* 为

$$H^p(\mathscr{T}_h) := \{v \in L_2(\Omega) : v|_\tau \in H^p(\tau) \textit{ for all } \tau \in \mathscr{T}_h\}$$

有分片 p 阶多项式组成的离散函数空间为

$$S_h^p(\mathscr{T}_h) := \{v_h \in L_2(\Omega) : v_h|_\tau \in \mathbb{P}_p(\tau), \forall \tau \in \mathscr{T}_h\}$$

限制在 $\tau \in \mathscr{T}_h$ 属于索伯列夫空间并且在 τ 的内部边界上可以不连续。

2.1 Possion 方程

考虑有混合边界的 Possion 方程，介绍其理论框架。首先 Possion 方程有如下形式：

$$\begin{cases} -\nabla \cdot (a(\nabla u)) = f & x \in \Omega \\ u = g_D & x \in \Gamma_D \\ u = g_N & x \in \Gamma_N \end{cases} \tag{2-1}$$

在这里我们假设 $g_D = 0$. 我们将其画为弱解形式: 对于 $\forall v \in V := \{v|_\tau \in H^2(\tau), \forall \tau \in \mathscr{T}_h\}$

$$\begin{aligned} (f, v) &= -\int_\Omega \nabla \cdot (a\nabla u)v = -\sum_{\tau \in \mathscr{T}_h} \int_\tau \nabla \cdot (a\nabla u)v \\ &= \sum_{\tau \in \mathscr{T}_h} \int_\tau a\nabla u \cdot \nabla v - \sum_{\tau \in \mathscr{T}_h} \int_{\partial\tau} a\mathbf{n}_\tau \cdot \nabla u v. \end{aligned} \tag{2-2}$$

先补充一个定理：

Theorem 1. ***(Sobolev Embedding Theroem)*** 若 $u \in W^{k,p}(\Omega)$, 其中 $p > 1$, 当 $m := k - \frac{n}{p} > 0$, 那么 $u \in C^m(\Omega)$

因此由上述定理，如果 $u \in H^2(\Omega)$, 所以 $u \in C^m(\Omega)$，即 $[u] = [\nabla u] = 0$

因此式子 (2-2) 的右边第二项可以写成：

$$\begin{aligned} \sum_{\tau \in \mathscr{T}_h} \int_{\partial\tau} a\nabla u \cdot \mathbf{n}_\tau v &= \sum_{e \in \mathcal{E}_h^I} \int_e [a\mathbf{n} \cdot \nabla u v] + \sum_{e \in \mathcal{E}_h^D} \int_e a\mathbf{n} \cdot \nabla u v + \sum_{e \in \mathcal{E}_h^N} \int_e a\mathbf{n} \cdot \nabla u v \\ &= \sum_{e \in \mathcal{E}_h^I} \int_e ([a\nabla u \cdot \mathbf{n}]\{v\} + \{a\nabla u \cdot \mathbf{n}\}[v]) + \sum_{e \in \mathcal{E}_h^D} \int_e a\nabla u \cdot \mathbf{n} v + \int_{\Omega_N} g_N v \\ &= \langle \{a\nabla u \cdot \mathbf{n}\}, [v] \rangle_{\mathcal{E}_h^{ID}} + \int_{\Omega_N} g_N v \end{aligned}$$

所以 (2-2) 可以有形式：

$$(a\nabla u, \nabla v)_{\mathscr{T}_h} - \langle \{a\nabla u \cdot \mathbf{n}\}, [v] \rangle_{\mathcal{E}_h^{ID}} = (f, v)_{\mathscr{T}_h} + \int_{\Omega_N} g_N v, \quad \forall v \in V. \tag{2-3}$$

为了最后刚度矩阵对称性考虑，添加对称项，对称系数为 ϵ。同时我们可以发现 (2-3) 中其实没有涉及边界条件，因此添加用于约束边界的惩罚项，惩罚因子为 σ。

于是我们可以定义双线性型：

$$\begin{cases} A_h(u,v) := (a\nabla u, \nabla v)_{\mathscr{T}_h} - \left(\langle \{a\mathbf{n}\cdot\nabla u\}, [v] \rangle_{\mathcal{E}_h^{ID}} - \epsilon \langle \{a\mathbf{n}\cdot\nabla v\}, [u] \rangle_{\mathcal{E}_h^{ID}} \right) + J_0(u,v) + J_1(u,v) \\ F_h(v) = (f,v)_{\mathscr{T}_h} + \displaystyle\int_{\Omega_N} g_N v + \epsilon \langle \{a\mathbf{n}\cdot\nabla v\}, [u] \rangle_{\mathcal{E}_h^D}, \quad \forall v \in V \end{cases}$$

其中

$$J_0(u,v) := \sum_{e\in\mathcal{E}_h^{ID}} \frac{\sigma_0}{h_e} \int_e [u][v], \qquad J_1(u,v) := \sum_{e\in\mathcal{E}_h^{I}} \sigma_1 h_e \int_e [a\nabla u\cdot n][a\nabla v\cdot n]$$

注意到对于精确解 u 而言，我们自然的有

$$\begin{cases} \epsilon \langle \{a\mathbf{n}\cdot\nabla v\}, [u] \rangle_{\mathcal{E}_h^{ID}} = \epsilon \langle a\mathbf{n}\cdot\nabla v, u \rangle_{\mathcal{E}_h^{D}} \\ J_0(u,v) = \displaystyle\sum_{e\in\mathcal{E}_h^{D}} \frac{\sigma_0}{h_e} \int_e g_D v \\ J_1(u,v) = 0 \end{cases}$$

因此我们需要找到 $u_h \in S_h^p(\mathscr{T}_h)$ s.t.

$$A_h(u_h, v_h) = F_h(v_h) \quad \forall v_h \in S_h^p(\mathscr{T}_h). \tag{2-4}$$

这里我们仅考虑拥有齐次边界条件的精确解 u 来说, 我们显然有

$$\langle \{a\nabla v\cdot n\}, [u] \rangle_{\mathcal{E}_h^{ID}} = J_0(u,v) = J_1(u,v) = 0, \quad \forall v \in V$$

之后我们只需要像平常的有限元方法一样寻找基函数即可。例如我们设 $u_h = \sum_{i=1}^{N_h} u_i \phi_i$, 那么

$$\sum_{j=1}^{N_h} A_h(\phi_j, \phi_i) u_j = F_h(\phi_i), i = 1 \cdots N_h$$

这可以被写作为矩阵形式 $AU = F$

因此上边所离散的问题等价于一个线性方程组的问题，我们可以通过选择合适的方法解决。并且可以注意到矩阵 A 为稀疏矩阵，当单元数量变多时，网格大小变为 $(Nloc \times Nelt)^2$。

注：

在这里考虑到矩阵的稀疏性，其实有一些工作针对与稀疏矩阵的处理，无论是经典的数值算法还是深度学习方法。

经典的数值算法例如多重网格方法用来加速矩阵运算。深度学习方法近些年有些工作专注于利用图神经网络进行加速求解，因为图结构所构成的邻接矩阵同样是大型稀疏的，之间非常相似。那么如何将图神经网络和有限元中的稀疏关系结合从而相辅相成是一个值的思考的问题。

接下来我们考虑关于求解瞬态 PDE 方程的特殊方法。

3 瞬态问题 (Time-dependent problems)

关于稳态方程，也即时间无关的偏微分方程，我们在很大程度上有很多手段处理。但是一旦涉及到时间项，如何处理解随时间动态变化的问题开始变得棘手起来。并且时间项处理的好坏直接影响着结果的优劣，一些经经常出现的问题如误差累积，以及并行计算的困难等等。

事实上我们主要有三种方法解决瞬态问题。

- 对于时间项，使用差分方法，如单步法，线性多步法，来进行时间步长迭代。
- 空间维度用有限元方法进行离散，最后将 PDE 系统化为为 ODE 系统，之后解关于 ODE 系统方程组。
- 利用时空有限元方法 (STFEM) 同时离散空间和时间维度;

基于 NS 方程在下面几部分中分别讨论了这三种方法。在这里，我们仅考虑其次 Dirichlet 边界条件，Ω 为空间区域，$I = (0, T)$ 为时间域，方程可以写作为:

$$\begin{cases} \mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} + \dfrac{1}{\rho}\nabla p - \gamma \Delta \mathbf{u} = \mathbf{f} \quad \text{in } \Omega \times I \\ \nabla \cdot \mathbf{u} = 0 \\ \mathbf{u}|_{\partial\Omega} = 0 \end{cases}$$

3.1 时间差分

首先我们尝试时间差分离散化。我们通常有高效并且成熟的数值方法来处理时间项，如龙格库塔方法。以向前欧拉方法 (Forward Euler Method) 为例子，我们有

$$\begin{aligned}&\frac{\mathbf{u}_{n+1}-\mathbf{u}_n}{\Delta t}=\gamma\Delta\mathbf{u}_n-(\mathbf{u}_n\cdot\nabla)\mathbf{u}_n-\frac{1}{\rho}\nabla p_n+\mathbf{f}\overset{\text{denote}}{=}\mathbf{F}(\mathbf{u}_n,p_n)\\&\Rightarrow\mathbf{u}_{n+1}=\mathbf{u}_n+\Delta t\mathbf{F}(\mathbf{u}_n,p_n)\end{aligned}\tag{3-1}$$

其中 $\mathbf{u}_n$ 代表在第 n 步的解 $\mathbf{u}$, 可以看到该方法主要以迭代的方式进行时间步长的推演。当然也有其他时间离散方案如 leapfrog method, RK method 等等，分为显式和隐式方法。

然而由于 $\mathbf{u}_{n+1}$ 可能不满足 $\nabla\cdot\mathbf{u}=0$ 并且没有办法处理压力项 p_n. 因此对于 NS 方程需要引入 Splitting Operator Method。方法如下：

通常我们同时有

$$\frac{\mathbf{u}_{n+1}-\mathbf{u}_n}{\Delta t}=\gamma\Delta\mathbf{u}_n-(\mathbf{u}_n\cdot\nabla)\mathbf{u}_n-\frac{1}{\rho}\nabla p_{n+1}+\mathbf{f}=\mathbf{F}(\mathbf{u}_n,p_{n+1})\tag{3-2}$$

其中 $\nabla\cdot\mathbf{u}_{n+1}=0$. 并且我们记 (3-1) 中 $\mathbf{u}_{n+1}$ 为 $\mathbf{u}^*$.

那么我们需要寻找一个修正 $\delta\mathbf{u}$ s.t. $\mathbf{u}_{n+1}=\mathbf{u}^*+\delta\mathbf{u}$。

因此

$$\delta\mathbf{u}=-\frac{\Delta t}{\rho}\nabla(p_{n+1}-\beta p_n)\overset{denote}{=}-\frac{\Delta t}{\rho}\nabla\Phi$$

i.e. $\Phi=p_{n+1}-p_n$.

代入 $\nabla\cdot\mathbf{u}_{n+1}=\nabla\cdot(\mathbf{u}^*+\delta\mathbf{u})=\nabla\cdot(\mathbf{u}^*-\frac{\rho}{\Delta t}\nabla\Phi)=0$, 也即

$$\Delta\Phi=\frac{\rho}{\Delta t}\nabla\cdot\mathbf{u}\tag{3-3}$$

然后我们可以更新 $\mathbf{u}_{n+1},p_{n+1}$，通过

$$\begin{cases}\mathbf{u}_{n+1}=\mathbf{u}^*-\dfrac{\Delta t}{\rho}\nabla\Phi\\[2ex]p_{n+1}=\Phi+\beta p_n\end{cases}\tag{3-4}$$

这是一种将复杂问题转化为简单的小问题的方法。

总结步骤如下:

- 计算 $\mathbf{u}^*$
- 解一般 Possion 方程 (3-3)
- 根据 (3-4) 更新 $\mathbf{u}_{n+1}$&p_{n+1}

随后我们推导弱解形式。引入测试函数 $\mathbf{v}^{(u)} \in V^{(u)}$& $\mathbf{v}^{(\phi)} \in V^{(\phi)}$。由分部积分可以得到如何两项的逼近形式:

$$\int_\Omega \Delta \mathbf{u}\mathbf{v} = -\int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} + \int_{\partial\Omega} \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \cdot \mathbf{v} ds$$
$$\int_\Omega \nabla p \cdot \mathbf{v} = -\int_\Omega p \cdot \nabla \cdot \mathbf{v} + \int_{\partial\Omega} p \cdot \mathbf{n}\mathbf{v} ds$$

其中 $\frac{\partial \mathbf{u}}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla \mathbf{u}$。因此我们得到弱解形式为

$$\begin{aligned}
&\int_\Omega \left[\mathbf{u}^* \mathbf{v}^{(u)} + \Delta t (\mathbf{u}_n \cdot \nabla)\mathbf{u}_n \cdot \mathbf{v}_n\right] - \int_\Omega \beta \frac{\Delta t}{\rho} p^n \nabla \cdot \mathbf{v}^{(u)} + \int_\Omega \Delta t \gamma \nabla \mathbf{u}_n : \nabla \mathbf{v}_n \\
&= \int_\Omega \Delta t \gamma \mathbf{f} \cdot \mathbf{v} + \int_\Omega \left(\Delta t \gamma \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \mathbf{v}^{(u)} - \beta \frac{\Delta t}{\rho} p \cdot \mathbf{n}\mathbf{v}\right) ds \\
&\int_\Omega \nabla \Phi \cdot \nabla \mathbf{v}^\phi = -\frac{\rho}{\Delta t} \int_\Omega \nabla \cdot \mathbf{u}^* \cdot \mathbf{v}^{(\phi)} + \int_{\partial\Omega} \frac{\partial \Phi}{\partial \mathbf{n}} \mathbf{v}^{(\phi)} ds
\end{aligned} \tag{3-5}$$

但是这种方法的缺陷是容易造成误差累计，并且需要很强的稳定性条件。因此我们需要更多的连续性，更好的离散性。

3.2 转化为 ODE 系统

我们可以直接得到 NS 方程的弱解形式:

$$\begin{cases}
\int_\Omega \mathbf{u}_t \mathbf{v} + \int_\Omega \gamma \nabla \mathbf{u} : \nabla \mathbf{v} + \frac{1}{\rho} \int_\Omega p \nabla \cdot \mathbf{v} + \int_\Omega (\mathbf{u} \cdot \nabla)\mathbf{u} \cdot \mathbf{v} \\
= \int_\Omega \mathbf{f} \cdot \mathbf{v} + \int_\Omega \left(\gamma \frac{\partial \mathbf{u}}{\partial \mathbf{n}} \mathbf{v}^{(u)} - \frac{1}{\rho} p \cdot \mathbf{n}\mathbf{v}\right) ds & \mathbf{v} \in V \\
\int_\Omega \nabla \cdot \mathbf{u} q = 0 & q \in W
\end{cases}$$

用记号可以化简为

$$\begin{cases}(\mathbf{u}_t,\mathbf{v})+\gamma a(\mathbf{u},\mathbf{v})-\dfrac{1}{\rho}b(\mathbf{v},p)+((\mathbf{u}\cdot\nabla)\mathbf{u},\mathbf{v})=(\mathbf{f},\mathbf{v})\\ b(\mathbf{u},q)=0\end{cases}$$

设离散化函数空间 $V_h\subset V, W_h\subset W$ 的基函数分别为为 $\{\phi_1,\phi_2,\cdots\phi_{N_v}\}$，因此可以设

$$u^h=\sum_{i=1}^{N_v}U(t,\mathbf{x}_i)\phi_i(\mathbf{x})$$

代入到弱形式中可以得到

$$\begin{cases}(\mathbf{u}_t^h,\mathbf{v}^h)+\gamma a(\mathbf{u}^h,\mathbf{v}^h)-\dfrac{1}{\rho}b(\mathbf{v}^h,p^h)+((\mathbf{u}^h\cdot\nabla)\mathbf{u}^h,\mathbf{v}^h)=(\mathbf{f},\mathbf{v}^h)\\ b(\mathbf{u}^h,q^h)=0\end{cases}$$

那么可以将方程转化为 ODE 系统。但是我们可以发现非线性项非常难处理。需要用类牛顿迭代等方法进行迭代计算。因此我们提出第三种方法时空方法。

4 时空有限元 (Space-Time FEM)

第三种方法研究了时空有限元方法[3]。这里的时空方法主要是指将时间维度作为最后一维加入到定义域中。这种方法最大的一个优点是他们可以自然的处理一些有关动边界的问题。另一个优点是在处理有局部奇点问题有很大优势。缺点也很容易发现，很容易陷入维度灾难。当问题涉及到三维之后，添加时间维度之后，定义域变成四维空间，数值计算变得非常复杂。但这提供了一种不同的看待时间维度的方法。

在这里我们首先主要介绍基于间断有限元的时空方法，并给出相关的误差以及收敛性分析。简单点来说，它和普通的有限元方法相比最大不同，就在于它的目标是寻找一组依赖于时间的基函数 $\phi_i(t,x)$ 的线性组合。首先基于热方程，介绍时空有限元方法的方案原理，随后基于 NS 方程再给出方案的理论分析。

4.1 热方程

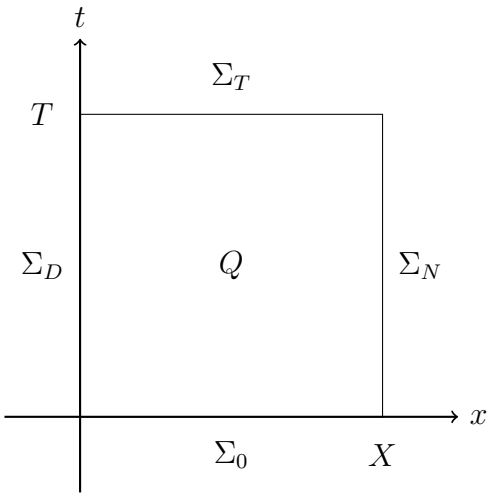
第一个模型我们基于拥有混合边界条件的热方程，其数学形式如下所示：

$$\begin{cases} cu_t(\mathbf{x},t) - \nabla_x \cdot [A(\mathbf{x},t)\nabla_x u(x,t)] = f(\mathbf{x},t) & \text{for } (\mathbf{x},t) \in Q := \Omega \times (0,T), \\ u(\mathbf{x},t) = g_D & \text{for } (\mathbf{x},t) \in \Sigma_D := \Gamma_D \times (0,T), \\ \mathbf{n}_x \cdot \nabla_x u = g_N & \text{for } (\mathbf{x},t) \in \Sigma_N := \Gamma_N \times (0,T) \\ u(\mathbf{x},0) = u_0(\mathbf{x}) & \text{for } \mathbf{x} \in \Omega, \end{cases}$$

其中 $A(\mathbf{x},t) = [A(\mathbf{x},t)]^\top \in \mathbb{R}^{n\times n}$ 为系数，在 $(\mathbf{x},t) \in Q$ 为正定矩阵形式，通常情况下我们将其设为 I; $c > 0$ 为热容常数，通常将其当作常数 1, T 为给定的模拟结束时间。$f \in L_2(0,T)$ 是给定的热源项。我们假设 $u_0 \in H^1(\Omega)$.

时空区域离散

记 Q(1D) 为 2D 时空区域，即空间区域为 1 维，时间作为第二维度，如下图所示



在这里我们只考虑纯 Dirichlet 边界条件。

这里的离散方法和 DG 方法种类似，但其实可以对区域进行广义离散，如图4.1，这在处理动边界问题时有很大的优势。

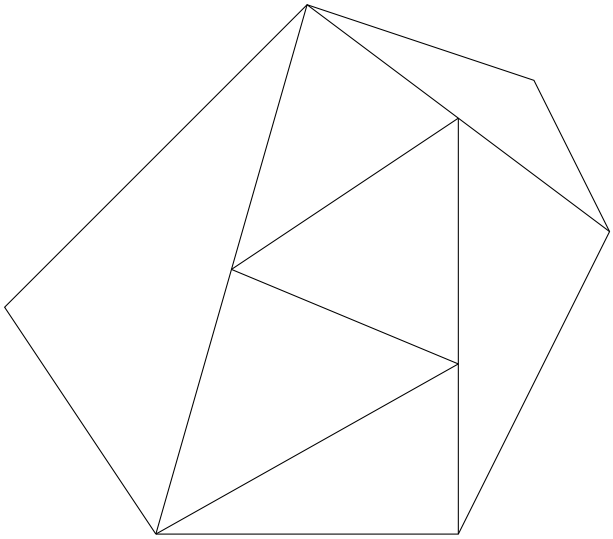


图 4.1: 广义离散

对于一般的时空间断有限元方法，通常采取的离散化方案是基于 Space-Time Slabs，如图4.2。但在我们这项工作中我们允许对时空区域做非结构化分解，允许在时空区域以任意单元形式离散化。

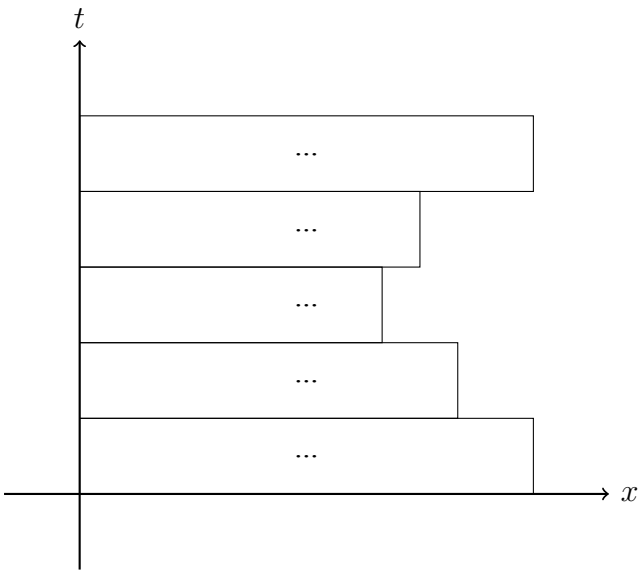


图 4.2: Space-Time Slabs

我们将时空区域记作 Q，其离散化记作 Q_h，并且 Q_h 中最简单元记作 τ_l, i.e.

$$Q_h = \bigcup_{l=1}^{N} \tau_l$$

可以发现当 n = 1 时，单元为三角形，n=2 时为四面体，n = 3 时为 Pentatops。我们以 2D(n=1) 为例，离散化区域如下:

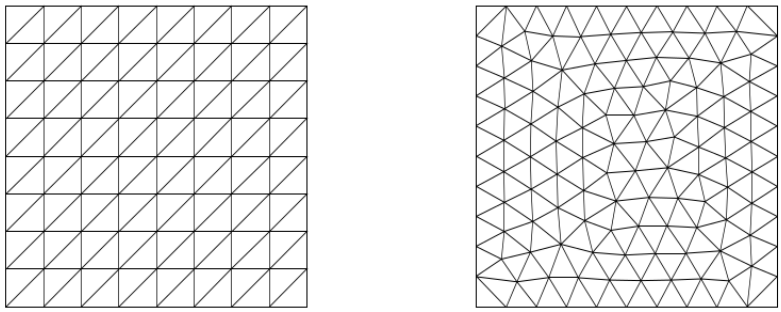


图 4.3: 结构化 (规则) 单元和非结构化 (不规则) 单元

一些定义

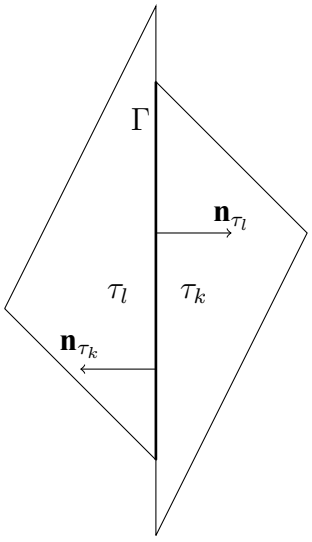
在推导变分公式之前，我们有如下一些定义：

Definition 3 (内切面 (Interior facet))**.** 对于相邻的两个单元 $\tau_k, \tau_l \in Q_h$ 来说, 内切面 Γ_{kl} 定义为

$$e_{kl} := \overline{\tau_k} \cap \overline{\tau_l}$$

并且我们把内切面的所有集合记作 $\mathscr{E}_h^I$。

自然可以发现当时空区域为二维时，内切面为一条线段；当时空区域为三维时，内切面为一个三维平面。这里二维内切面如下所示：



其中 $\mathbf{n}_{\tau_l}$ 为单元 τ_l 向外的单位法向量，$\mathbf{n}_{\tau_k}$ 为单元 τ_k 向外的单位法向量。

随后我们有一些和间断有限元相似的定义，但仍然有些不同：

Definition 4 (Jump and average)**.** .

跳量和平均 *1.* 在内切面 e_{kl} 上的跳量定义为:

$$[v]_{e_{kl}} := v|_{\tau_k}\boldsymbol{n}_k + v|_{\tau_l}\boldsymbol{n}_l$$

更具体来说，在内切面 e_{kl} 上空间方向的跳量定义为

$$[v]_{e_{kl},\boldsymbol{x}} := v|_{\tau_k}n_{k,\boldsymbol{x}} + v|_{\tau_l}n_{l,\boldsymbol{x}},$$

时间方向的跳量为：

$$[v]_{e_{kl},t} := v|_{\tau_k}n_{k,t} + v|_{\tau_l}n_{l,t}.$$

2. 函数 v 在内切面 e_{kl} 上定义的平均量为：

$$\langle v\rangle_{e_{kl}} := \frac{1}{2}\left(v|_{\tau_k} + v|_{\tau_l}\right),$$

3. 在时间方向的迎风量为:

$$\{v\}^{up}_{e_{kl}} := \begin{cases} v|_{\tau_k} & \textit{for } n_{k,t} > 0 \\ 0 & \textit{for } n_{k,t} = 0 \\ v|_{\tau_l} & \textit{for } n_{k,t} < 0 \end{cases}.$$

4. 时间方向的顺风量为:

$$\{v\}^{down}_{e_{kl}} := \begin{cases} v|_{\tau_l} & \textit{for } n_{k,t} > 0 \\ 0 & \textit{for } n_{k,t} = 0 \\ v|_{\tau_k} & \textit{for } n_{k,t} < 0 \end{cases}.$$

在这里 $\mathbf{n}_k$ 为单元 τ_k 边上的单位法向量。其中 $n_x = \mathbf{n}_k \cdot \hat{x}, n_t = \mathbf{n}_k \cdot \hat{t}$，$\hat{x}, \hat{t}$ 分别代表空间和时间方向的单位向量。并且我们注意到对于在边界上的边 $e_{kl} \in \mathscr{E}_h^D, [v] = \{v\} = v$, 属于内部边的内切面 $e \in \mathscr{E}_h^I$, 有 $[vw] = [v]\{w\} + \{v\}[w]$

变分形式

我们有弱解形式如下：

$$\int_Q u_t v - \int_Q \Delta u v = \int_Q f v \tag{4-1}$$

因此我们用内罚间断伽辽金方法来逼近拉普拉斯算子 Δu，这里和利用 DGM 求解 Possion 方程类似，但由于时间维度的加入，推导变得复杂：

$$\begin{aligned} a\left(u_h, v_h\right) := & \sum_{l=1}^{N} \int_{\tau_l} \nabla_x u_h \cdot \nabla_{\boldsymbol{x}} v_h - \sum_{e_{kl} \in \mathscr{E}_h^{ID}} \int_{e_{kl}} \left\langle \nabla_{\boldsymbol{x}} u_h \right\rangle_{e_{kl}} \cdot \left[v_h\right]_{e_{kl}, \boldsymbol{x}} \mathrm{d}s \\ & + \epsilon \sum_{e_{kl} \in \mathscr{E}_h^{ID}} \int_{e_{kl}} \left[u_h\right]_{e_{kl}, \boldsymbol{x}} \cdot \left\langle \nabla_{\boldsymbol{x}} v_h \right\rangle_{e_{kl}} \mathrm{d}s \\ & + \sum_{e_{kl} \in \mathscr{E}_h^{ID}} \frac{\sigma}{\overline{h_{kl}}} \int_{e_{kl}} \left[u_h\right]_{e_{kl}, \boldsymbol{x}} \cdot \left[v_h\right]_{e_{kl}, \boldsymbol{x}} \mathrm{d}s \end{aligned} \tag{4-2}$$

注意第三项和第四项分别为添加的对称项和惩罚项。

对于时间导数，我们由格林公式可得:

$$\int_{\partial \tau} u v \cdot n_t ds = \int_{\tau} (u_t v + u v_t) d\tau$$

注意到在 Σ_D 上 $n_t = 0$, 那么我们可以逼近 u_t 通过

$$b\left(u_h, v_h\right) := -\sum_{l=1}^{N} \int_{\tau_l} u_h \partial_t v_h + \int_{\Sigma_T} u_h v_h \cdot n_t ds + \sum_{e_{kl} \in \mathscr{E}_h^{I}} \int_{e_{kl}} \left\{u_h\right\}_{e_{kl}}^{\mathrm{up}} \left[v_h\right]_{e_{kl}, t} ds \tag{4-3}$$

因此由 IPDG, 问题可以转化为:

寻找 $u_h \in S_h^p(Q_h)$ s.t.

$$A\left(u_h, v_h\right) = \left\langle f, v_h \right\rangle_Q + \left\langle u_0, v_h \right\rangle_{\Sigma_0} + \left\langle g, v_h \right\rangle_{\Sigma_N}$$

对于任意的 $v_h \in S_h^p\left(Q_h\right)$.

其中 $A\left(u_h, v_h\right) := a\left(u_h, v_h\right) + b\left(u_h, v_h\right)$

那么我们可以定义 $S_h^p(Q_h)$ 的基函数 $span\{\phi_k\}_{k=1}^M$，i.e.

$$S_h^p(Q_h) = span\{\phi_k\}_{k=1}^M, u_h(x,t) = \sum_{j=1}^{M} u_j\phi_j(x,t) \qquad u_h \in S_h^p(Q_h)$$

可以写成矩阵形式

$$A_h u = f$$

其中 $A_h = A(\phi_j, \phi_i), f_i = \langle f, \phi_i \rangle_Q + \langle u_0, \phi_i \rangle_{\Sigma_0} + \langle g, \phi_i \rangle_{\Sigma_N}$

5 STFEM for NS

在热方程之后，我们再考虑 NS 方程。因此首先利用混合有限元方法进行解耦速度和压力。

5.1 混合有限元方法 (Mixed Finite Element Method)

我们在变量耦合的地方应用 MFEM。斯托克斯方程是 NS 方程雷诺数趋向于 0 的特殊情况，因此我们以斯托克斯方程作为例子。

$$\begin{cases} -\Delta \mathbf{u} + \nabla p = \mathbf{f} & in\ \Omega \\ u = 0 & on\ \Gamma_D \\ \nabla \cdot \mathbf{u} = 0 & \Gamma_N \end{cases}$$

其中 $\mathbf{u} = (u_1, u_2)$ 为速度场中向量, p 为压力场为标量, $\mathbf{f} = (f_1, f_2)$。这里的困难在与如何解耦 u 和 p 这两个变量。为了解的唯一性，我们需要施加额外条件 $\int_\Omega p = 0$

因此我们可以得到斯托克斯方程的弱形式：寻找 $\mathbf{u} \in V = [H_0^1]^2$ 并且 $p \in W = L_0^2$ s.t.

$$\begin{cases} a(\mathbf{u}, \mathbf{v}) - b(\mathbf{v}, p) = (\mathbf{f}, \mathbf{v}) & \forall \mathbf{v} \in V \\ b(\mathbf{u}, q) = 0 & \forall q \in W \end{cases}$$

其中

$$a(\mathbf{u}, \mathbf{v}) = (\nabla \mathbf{u}, \nabla \mathbf{v}) = \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} d\Omega$$

$$b(\mathbf{v}, q) = (\nabla \cdot \mathbf{v}, q) = \int_\Omega \nabla \cdot \mathbf{v} \cdot q$$

5.1.1 优化

我们可以把上述问题转化为一个优化问题

$$\begin{aligned}\mathbf{u} &= arg \min_{\mathbf{v}\in V, \nabla\cdot\mathbf{v}=0} \left\{ \frac{1}{2} \|\nabla \mathbf{v}\|^2_{L^2(\Omega)} - (\mathbf{f}, \mathbf{v}) \right\} \\ &= arg \min_{\mathbf{v}\in V} \max_{q\in W} \left\{ \frac{1}{2} \|\nabla \mathbf{v}\|^2_{L^2(\Omega)} - (\mathbf{f}, \mathbf{v}) - (\nabla\cdot\mathbf{v}, q) \right\}\end{aligned}$$

注：

这里 min-max 的形式自然形成对抗的形式，很自然的可以参考弱对抗神经网络 (WAN) 进行处理。因此我认为弱对抗神经网络是是处理弱解形式耦合问题好的方法 (原本的弱对抗神经网络的关注点在于处理高维 PDE)。

5.1.2 离散化

我们引入 $V_h \subset V, W_h \subset W$ 并且想要寻找 V_h 基函数 $\{\phi_1, \phi_2 \cdots \phi_n\}$, W_h 中基函数 $\{\psi_1, \psi_2 \cdots \psi_m\}$. 因此我们需要

寻找 $(\mathbf{u}_h, p_h) \in V_h \times W_h$, 更具体的是, $\mathbf{u}_h = \sum_i \mathbf{U}(\mathbf{x}_i)\phi_i$ 和 $p = \sum_i P(\mathbf{x}_i)\psi_i$ s.t.

$$\begin{cases} a(\mathbf{u}^h, \mathbf{v}^h) - b(\mathbf{v}^h, p^h) = (\mathbf{f}, \mathbf{v}) & \forall \mathbf{v}^h \in V_h \\ b(\mathbf{u}^h, q^h) = 0 & \forall q \in W^h \end{cases}$$

因此我们得到

$$\begin{cases} AU - BP = F \\ B^T U = 0 \end{cases} \quad \text{i.e.} \quad \begin{pmatrix} A & -B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix}$$

其中 $A = (a_{ij}), a_{ij} = a(\phi_i, \phi_j) = (\nabla\phi_i, \nabla\phi_j); B = (b_{ij}), b_{ij} = b(\phi_j, \psi_i) = (\psi_i, \nabla\cdot\phi_j); F = (F_i), F_i = (\mathbf{f}, \phi_i)$

这是解决斯托克斯方程的混合有限元方法，它一个和时间无关的静态方程，然而对于真实的 NS 方程是瞬态方程。所以如何处理时间变量是一个重要的问题，这在上一节已经讨论过两种方法，下面展示时空方法。

5.2 STDGM for NS

在考虑热方程之后，我们考虑 NS 方程的时空有限元方法。首先有方程：

$$\begin{cases} \partial_t \boldsymbol{u} - v\Delta \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla_{\boldsymbol{x}})\,\boldsymbol{u} + \nabla_{\boldsymbol{x}} p = \boldsymbol{f} & \text{in } Q, \\ \nabla\cdot(\boldsymbol{u}) = 0 & \text{in } Q, \\ \boldsymbol{u} = \boldsymbol{g}_D & \text{on } \Sigma_D, \\ v\,(\nabla_{\boldsymbol{x}}\boldsymbol{u})\,\boldsymbol{n}_x - p\boldsymbol{n}_x = \boldsymbol{0} & \text{on } \Sigma_N, \\ \boldsymbol{u} = \boldsymbol{u}_0 & \text{on } \Sigma_0. \end{cases}$$

定义函数空间

$$\begin{cases} V_h^{p+1}(Q_h) := \left[S_h^{p+1}(Q_h)\right]^d = \left\{\boldsymbol{v}_h \in [L_2(Q)]^d : \boldsymbol{v}_h|_{\tau_l} \in [\mathbb{P}_{p+1}(\tau_l)]^d \text{ for all } \tau_l \in Q_h, \boldsymbol{v}_h = \boldsymbol{0} \text{ on } \Sigma_D\right\} \\ W_h^p(Q_h) := \left\{q_h \in L_2(Q) : q_h|_{\tau_l} \in \mathbb{P}_p(\tau_l) \text{ for all } \tau_l \in Q_h\right\}. \end{cases}$$

首先式子中有热方程中的两项 $\partial_t\boldsymbol{u} - v\Delta\boldsymbol{u}$, 可以利用上述热方程的推导过程化为 $A(u_h, v_h) := a(u_h, v_h) + b(u_h, v_h)$, 这里和热方程一样。

接下来我们考虑如何处理压力项的梯度 $\nabla_x p$。我们同样有弱形式，并通过分部积分写成跳量形式可以得到：

$$\begin{aligned} & -\int_Q \nabla_{\boldsymbol{x}} p\cdot\boldsymbol{v}_h \\ =& -\sum_{\ell=1}^N \int_{\tau_\ell} \nabla_{\boldsymbol{x}} p\cdot\boldsymbol{v}_h = \sum_{\ell=1}^N \int_{\tau_\ell} p\nabla_x\cdot v_h - \sum_{\ell=1}^N \int_{\partial\tau_\ell} p\boldsymbol{v}_h\cdot\boldsymbol{n}_{\boldsymbol{x}}\mathrm{d}s \\ =& \sum_{\ell=1}^N \int_{\tau_\ell} p\nabla_x\cdot v_h - \int_{\Sigma_N} p\boldsymbol{v}_h\cdot\boldsymbol{n}_{\boldsymbol{x}}\mathrm{d}s - \langle\boldsymbol{u}_0, \boldsymbol{v}_h\rangle_{\Sigma_0} - \sum_{\Gamma_{k\ell}\in\mathscr{E}_h^I}\int_{\Gamma_{k\ell}} [p\boldsymbol{v}_h]_{\Gamma_{k\ell},\boldsymbol{x}}\,\mathrm{d}s \\ =& \sum_{\ell=1}^N \int_{\tau_\ell} p\nabla_x\cdot v_h - \int_{\Sigma_N} p\boldsymbol{v}_h\cdot\boldsymbol{n}_{\boldsymbol{x}}\mathrm{d}s - \sum_{\Gamma_{k\ell}\in\mathscr{E}_h^I}\int_{\Gamma_{k\ell}} \langle p\rangle_{\Gamma_{k\ell}}[\boldsymbol{v}_h]_{\Gamma_{k\ell},\boldsymbol{x}}\mathrm{d}s - \langle\boldsymbol{u}_0, \boldsymbol{v}_h\rangle_{\Sigma_0} \end{aligned}$$

因此我们可以定义双线性型

$$B(v_h, p_h) = \sum_{\ell=1}^N \int_{\tau_\ell} p_h\nabla_x\cdot v_h - \sum_{\Gamma_{k\ell}\in\mathscr{E}_h^I}\int_{\Gamma_{k\ell}} \langle p_h\rangle_{\Gamma_{k\ell}}[\boldsymbol{v}_h]_{\Gamma_{k\ell},\boldsymbol{x}}\mathrm{d}s$$

根据混合有限元方法，可以得到 $B(u_h, q_h) = 0 \quad \forall q_h \in W_h^p(Q_h)$，并且用该式子代替 $\nabla \cdot u = 0$。

同时我们给压力项添加稳定项

$$D(p_h, q_h) := \sigma_p \sum_{\Gamma_{k\ell} \in \mathcal{I}_N} \bar{h}_{k\ell} \int_{\Gamma_{k\ell}} [p_h]_{\Gamma_{k\ell}}(\boldsymbol{x}, t) \cdot [q_h]_{\Gamma_{k\ell}}(\boldsymbol{x}, t) \mathrm{d}s_{(\boldsymbol{x},t)}$$

问题变成：

寻找 $\boldsymbol{u}_h = \boldsymbol{u}_{h,0} + \mathcal{E}\boldsymbol{g}_D$ 其中 $\boldsymbol{u}_{h,0} \in V_h^{p+1}(Q_h)$ 和 $p_h \in W_h^p(Q_h)$, s.t.

$$A(\boldsymbol{u}_h, \boldsymbol{v}_h) + \langle (\boldsymbol{u}_h \cdot \nabla_{\boldsymbol{x}}) \boldsymbol{u}_h, \boldsymbol{v}_h \rangle_Q - B(\boldsymbol{v}_h, p_h) = \langle \boldsymbol{f}, \boldsymbol{v}_h \rangle_Q + \langle \boldsymbol{u}_0, \boldsymbol{v}_h \rangle_{\Sigma_0},$$

$$B(\boldsymbol{u}_h, q_h) + D(p_h, q_h) = 0$$

对于任意 $\boldsymbol{v}_h \in V_h^{p+1}(Q_h), q_h \in W_h^p(Q_h)$

同时对于离散化函数空间，我们定义基函数

$$\begin{cases} V_h^{p+1}(Q_h) = \mathrm{span}\{\boldsymbol{\varphi}_{l=1}\}_l^{M_u}, & \boldsymbol{u}_{h,0} = \sum_{l=1}^{M_u} \boldsymbol{u}[l]\boldsymbol{\varphi}_l \text{ for } \boldsymbol{u}_{h,0} \in V_h^{p+1}(Q_h) \\ W_h^p(Q_h) = \mathrm{span}\{\psi_n\}_{n=1}^{M_p}, & p_h = \sum_{n=1}^{M_p} \boldsymbol{p}[n]\psi_n \text{ for } p_h \in W_h^p(Q_h) \end{cases}$$

基于此我们定义非线性算子

$$(K_h \boldsymbol{u})[k] := A(\boldsymbol{u}_h, \boldsymbol{\varphi}_k) + \langle (\boldsymbol{u}_h \cdot \nabla_{\boldsymbol{x}}) \boldsymbol{u}_h, \boldsymbol{\varphi}_k \rangle_Q$$

其中 $\boldsymbol{u}_h = \mathcal{E}\boldsymbol{g}_D + \sum_{l=1}^{M_u} \boldsymbol{u}[l]\boldsymbol{\varphi}_l$

以及 $B_h[m,l] := B(\varphi_l, \psi_m), \quad D_h[m,n] := D(\psi_n, \psi_m)$，其中 $k, l = 1, \ldots, M_u$，$m, n = 1, \ldots, M_p$.

可以得到矩阵形式

$$\begin{pmatrix} K_h & -B_h^\top \\ B_h & D_h \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_u \\ \boldsymbol{f}_p \end{pmatrix}$$

可以看到由于 u_h 未知，方程中的非线性项非常难处理,在这里我们可以通过牛顿迭代的方法处理。当然随着深度学习在各个领域的发展，在非线性优化方面也显现出很大的潜力。

6 关于多尺度问题

当离散化空间非常大的时候，有比较将区域进行分解，例如可以对于微观和宏观的区域可以各自进行不同的离散化策略。然而对于区域分解来说，困难在于边界处如何耦合的问题。因此在这里提出混合离散的方式，考虑针对有多尺度的问题。

6.1 混合离散

我们通过将区域分解成多个小区域，并且对于不同的区域可以采取不同的离散化方案。因此，对于无论是空间还是时间上的多尺度问题我们都可以通过区域上的网格加密来实现。

首先说明区域的分解方案。我们提出将时空区域 Q 分解为子区域 Q_i, i.e.$Q = \cup_i^p Q_i$。如图6.1所示, 且记 $\Sigma_i = \overline{\partial Q_i / \partial Q}, \Sigma = \cup_i^p \Sigma_i$ 可以看出 Σ 为不同的子区域的交界面。对于不同的子区域 Q_i 我们有不同的分解: $Q_i = \mathscr{T}_{N_i} := \cup_{l=1}^{N_i} \tau_l^i$.

因此有

$$\begin{cases} \overline{Q} = \overline{\mathscr{T}_N} = \cup_{i=1}^p \overline{\mathscr{T}_{N_i}} = \cup_i^p \cup_l^{N_i} \overline{\tau_l^i} \\ \Sigma_h = \mathscr{E}_N^I / \cup_i^p \mathscr{E}_{N_i}^I \end{cases}$$

在这里我们将 $\mathscr{E}_N^I$ 记作 $\mathcal{I}_N$.

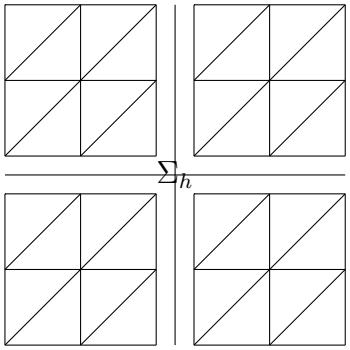


图 6.1: Subdomains

同时我们定义离散化函数空间

$$S_h^p(\Sigma_h) = \{v_h \in L_2(\Sigma) : v_h|_{\Gamma_{kl}} \in P_p(\Gamma_{kl}), \forall \Gamma_{kl} \in \Sigma_h\}$$

为了书写时下标不至于太多，在这里我们同样以前面简单的热方程为例，介绍其理论：

对于整个区域上写成弱形式，同之前介绍的理论，即寻找 $u_h \in S_h^p(\mathscr{T}_N)$ s.t.$\forall v_h \in S_h^p(\mathscr{T}_N)$,

$$A(u_h, v_h) = \langle f, v_h\rangle_Q + \langle u_0, v_h\rangle_{\Sigma_0} + \langle g_N, v_h\rangle_{\Sigma_N}$$

那么我们可以分别将该方案应用到 Q_i 上。因此我们有局部的双线性型: 其中 $u_h^i, v_h^i \in S_h^p(\mathscr{T}_{N_i})$ s.t. 对于 $i = 1, \ldots, P$

$$A^{(i)}(u_h^i, v_h^i) := a^{(i)}(u_h^i, v_h^i) + b^{(i)}(u_h^i, v_h^i) = F^{(i)}\left(v_h^i\right)$$

其中

$$\begin{cases} a^{(i)}(u_h^i, v_h^i) = \displaystyle\sum_{l=1}^{N_i} \int_{\tau_l^i} \nabla_x u_h^i \cdot \nabla_x v_h^i d\tau \\ \qquad\qquad - \displaystyle\sum_{\Gamma_{kl}\in\mathcal{I}_{N_i}} \int_{\Gamma_{kl}} \left(\left\langle \nabla_{\boldsymbol{x}} u_h^i \right\rangle_{\Gamma_{kl}} \cdot \left[v_h^i\right]_{\Gamma_{kl},\boldsymbol{x}} + \left[u_h^i\right]_{\Gamma_{kl},\boldsymbol{x}} \cdot \left\langle \nabla_{\boldsymbol{x}} v_h^i \right\rangle_{\Gamma_{kl}} \right) \mathrm{d}s \\ \qquad\qquad + \displaystyle\sum_{\Gamma_{kl}\in\mathcal{I}_{N_i}} \frac{\sigma}{\overline{h}_{kl}} \int_{\Gamma_{kl}} \left[u_h^i\right]_{\Gamma_{kl},\boldsymbol{x}} \cdot \left[v_h^i\right]_{\Gamma_{kl},\boldsymbol{x}} \mathrm{d}s \\ b^{(i)}\left(u_h^i, v_h^i\right) = -\displaystyle\sum_{l=1}^{N_i} \int_{\tau_l^i} u_h^i \partial_t v_h^i + \int_{\Sigma_T\cap\partial Q_i} u_h^i v_h^i \mathrm{d}s + \sum_{\Gamma_{kl}\in\mathcal{I}_{N_i}} \int_{\Gamma_{kl}} \left\{u_h^i\right\}_{\Gamma_{kl}}^{\mathrm{up}} \left[v_h^i\right]_{\Gamma_{kl},t} \mathrm{d}s \\ F^{(i)}\left(v_h^i\right) = \left\langle f, v_h^i\right\rangle_{Q_i} + \left\langle u_0, v_h^i\right\rangle_{\Sigma_0\cap\partial Q_i} + \left\langle g_N, v_h^i\right\rangle_{\Sigma_N\cap\partial Q_i}, \end{cases} \tag{6-1}$$

由于 $u_h^i = u_h|_{Q_i}$，因此我们可以将整个区域上的双线性型 $A(\cdot,\cdot)$ 分解成局部的双线性型 $A^{(i)}(\cdot,\cdot)$ 的和以及区域交界面 Σ_h 上的耦合部分。

即：

$$\begin{aligned} A\left(u_h, v_h\right) = & \sum_{i=1}^{P} A^{(i)}\left(u_h, v_h\right) - \sum_{\Gamma_{kl}\in\Sigma_h} \int_{\Gamma_{kl}} \left(\left\langle \nabla_{\boldsymbol{x}} u_h \right\rangle_{\Gamma_{kl}} \cdot \left[v_h\right]_{\Gamma_{kl},\boldsymbol{x}} + \left[u_h\right]_{\Gamma_{kl},\boldsymbol{x}} \cdot \left\langle \nabla_{\boldsymbol{x}} v_h \right\rangle_{\Gamma_{kl}} \right) \mathrm{d}s \\ & + \sum_{\Gamma_{kl}\in\Sigma_h} \frac{\sigma}{\overline{h}_{kl}} \int_{\Gamma_{kl}} \left[u_h\right]_{\Gamma_{kl},\boldsymbol{x}} \cdot \left[v_h\right]_{\Gamma_{kl},\boldsymbol{x}} \mathrm{d}s + \sum_{\Gamma_{kl}\in\Sigma_h} \int \left\{u_h\right\}_{\Gamma_{kl}}^{\mathrm{up}} \left[v_h\right]_{\Gamma_{kl},t} \mathrm{d}s. \end{aligned} \tag{6-2}$$

为了减少交界面 Σ 上的耦合量，在交界面 Σ 上定义如下新的变量：$\lambda_h \in S_h^p(\Sigma_h)$: $\lambda_h|_{\tau_{kl}} = \frac{1}{2}(u_h|_{\tau_k} + u_h|_{\tau_l})$。

同时我们有如下新记号：

Definition 5 (Hybrid Jump)**.**

$$[u/\lambda]_{\partial\tau_k} = (u|_{\tau_k} - \lambda)\boldsymbol{n}_k$$

那么自然的可以推导出如下性质

$$[u]_{\Gamma_{kl}} = 2[u/\lambda]_{\partial\tau_k} = 2[u/\lambda]_{\partial\tau_l}$$

且

$$[u_h]_{\Gamma_{kl},x} \cdot \langle \nabla_x v_h \rangle = [u_h/\lambda_h]_{\partial\tau_k,x} \cdot \nabla v_h|_{\tau_k} + [u_h/\lambda_h]_{\partial\tau_l,x} \cdot \nabla v_h|_{\tau_l}$$

因此

$$\begin{aligned}
&\sum_{\Gamma_{kl}\in\Sigma_h} \int_{\Gamma_{kl}} [u_h]_{\Gamma_{kl},\boldsymbol{x}} \cdot \langle \nabla_{\boldsymbol{x}} v_h \rangle_{\Gamma_{kl}} \,\mathrm{d}s \\
=&\sum_{\Gamma_{kl}\in\Sigma_h} \int_{\Gamma_{kl}} \left([u_h/\lambda_h]_{\partial\tau_k,\boldsymbol{x}} \cdot \nabla_{\boldsymbol{x}} v_h|_{\tau_k} + [u_h/\lambda_h]_{\partial\tau_l,\boldsymbol{x}} \cdot \nabla_{\boldsymbol{x}} v_h|_{\tau_l} \right) \mathrm{d}s \\
=&\sum_{i=1}^{P} \sum_{l=1}^{N_i} \sum_{\substack{\Gamma_{kl}\in\Sigma_h \\ \Gamma_{kl}\subset\partial\tau_l^i}} \int_{\Gamma_{kl}} [u_h/\lambda_h]_{\partial\tau_l^i,\boldsymbol{x}} \cdot \nabla_{\boldsymbol{x}} v_h|_{\tau_l^i} \mathrm{d}s.
\end{aligned}$$

同理，我们可以定义关于测试函数 v_h 的新变量 $\mu_h = \frac{1}{2}(v_h|_{\tau_k} + v_h|_{\tau_l}) \in S_h^p(\Sigma_h)$, 因此对称项变为

$$\sum_{\Gamma_{kl}\in\Sigma_h} \int_{\Gamma_{kl}} \langle \nabla_{\boldsymbol{x}} u_h \rangle_{\Gamma_{kl}} \cdot [v_h]_{\Gamma_{kl},\boldsymbol{x}} \,\mathrm{d}s = \sum_{i=1}^{P} \sum_{l=1}^{N_i} \sum_{\substack{\Gamma_{kl}\in\Sigma_h \\ \Gamma_{kl}\subset\partial\tau_l^i}} \int_{\Gamma_{kl}} \nabla_{\boldsymbol{x}} u_h|_{\tau_l^i} \cdot [v_h/\mu_h]_{\partial\tau_l^i,\boldsymbol{x}} \,\mathrm{d}s.$$

同时我们注意到

$$[u_h]_{\Gamma_{kl},\boldsymbol{x}} \cdot [v_h]_{\Gamma_{kl},\boldsymbol{x}} = 2\,[u_h/\lambda_h]_{\partial\tau_k,\boldsymbol{x}} \cdot [v_h/\mu_h]_{\partial\tau_k,\boldsymbol{x}} + 2\,[u_h/\lambda_h]_{\partial\tau_l,\boldsymbol{x}} \cdot [v_h/\mu_h]_{\partial\tau_l,\boldsymbol{x}}$$

因此惩罚项可以写成

$$\begin{aligned}
&\sum_{\Gamma_{kl}\in\Sigma_h} \frac{\sigma}{\overline{h}_{kl}} \int_{\Gamma_{kl}} [u_h]_{\Gamma_{kl},\boldsymbol{x}} \cdot [v_h]_{\Gamma_{kl},\boldsymbol{x}} \,\mathrm{d}s \\
=&\sum_{\Gamma_{kl}\in\Sigma_h} \frac{2\sigma}{\overline{h}_{kl}} \int_{\Gamma_{kl}} \left([u_h/\lambda_h]_{\partial\tau_k,\boldsymbol{x}} \cdot [v_h/\mu_h]_{\partial\tau_k,\boldsymbol{x}} + [u_h/\lambda_h]_{\partial\tau_l,\boldsymbol{x}} \cdot [v_h/\mu_h]_{\partial\tau_l,\boldsymbol{x}} \right) \mathrm{d}s \\
=&\sum_{i=1}^{P} \sum_{l=1}^{N_i} \sum_{\substack{\Gamma_{kl}\in\Sigma_h \\ \Gamma_{kl}\subset\partial\tau_l^i}} \frac{2\sigma}{\overline{h}} \int_{\Gamma_{kl}} [u_h/\lambda_h]_{\partial\tau_l^i,\boldsymbol{x}} \cdot [v_h/\mu_h]_{\partial\tau_l^i,\boldsymbol{x}} \,\mathrm{d}s.
\end{aligned}$$

关于在6-2中涉及迎风变量的项，我们可以同样定义

Definition 6 (Hybrid Upwind)**.**

$$\{u/\lambda\}^{\mathrm{up}}_{\partial\tau_k}(\boldsymbol{x},t) := \begin{cases} u|_{\tau_k}(\boldsymbol{x},t) & \textit{for } n_{k,t} \geq 0, \\ 0 & \textit{for } n_{k,t} = 0, \\ \lambda(\boldsymbol{x},t) & \textit{for } n_{k,t} < 0 \end{cases} \qquad \textit{for } (\boldsymbol{x},t) \in \Gamma_{kl}.$$

那么对于精确解我们有, $\lambda = \langle u \rangle_{\Gamma_{kl}} = u|_{\tau_k} = u|_{\tau_l}$, 则

$$\sum_{\Gamma_{kl}\in\Sigma_h} \int_{\Gamma_{kl}} \{u_h\}^{\mathrm{up}}_{\Gamma_{kl}} [v_h]_{\Gamma_{kl},t} \,\mathrm{d}s = \sum_{i=1}^{P} \sum_{l=1}^{N_i} \sum_{\substack{\Gamma_{kl}\in\Sigma_h \\ \Gamma_{kl}\subset\partial\tau_l^i}} \int_{\Gamma_{kl}} \{u_h/\lambda_h\}^{up}_{\partial\tau_l^i} [v_h/\mu_h]_{\partial\tau_l^i,x} ds$$

因此我们可以将6-2中所有涉及耦合边界上的项写为

$$\begin{aligned} & C^{(i)}(u_h,\lambda_h;v_h,\mu_h) \\ =& -\sum_{l=1}^{N_i} \sum_{\substack{\Gamma_{kl}\in\Sigma_h \\ \Gamma_{kl}\subset\partial\tau_l^i}} \int_{\boldsymbol{l}} \nabla_{\boldsymbol{x}l} u_{h|\tau_l^i} \cdot [v_h/\mu_h]_{\partial\tau_l^i,\boldsymbol{x}} \,\mathrm{d}s \\ &+ \sum_{l=1}^{N_i} \sum_{\substack{\Gamma_{kl}\in\Sigma_h \\ \Gamma_{kl}\subset\partial\tau_l^i}} \frac{2\sigma}{\overline{h}_{kl}} \int_{\Gamma_{kl}} [u_h/\lambda_h]_{\partial\tau_l^i,\boldsymbol{x}} \cdot [v_h/\mu_h]_{\partial\tau_l^i,\boldsymbol{x}} \,\mathrm{d}s \\ &+ \sum_{l=1}^{N_i} \sum_{\substack{\Gamma_{kl}\in\Sigma_h \\ \Gamma_{kl}\subset\partial\tau_l^i}} \int_{kl} \{u_h/\lambda_h\}^{\mathrm{up}}_{\partial\tau_l^i} [v_h/\mu_h]_{\partial\tau_l^i,\boldsymbol{x}} \,\mathrm{d}s. \end{aligned}$$

最终我们将问题转化为：

求解 $u_h \in S_h^p(\mathscr{T}_h) \& \lambda_h \in S_h^p(\Sigma_h)$ s.t. for $\forall v_h \in S_h^p(\mathscr{T}_h) \& \mu_h \in S_h^p(\Sigma_h)$,

$$\sum_{i=1}^{p} \left[A^{(i)}(u_h,v_h) + C^{(i)}(u_h,\lambda_h;v_h,\mu_h) \right] = \sum_{i=1}^{p} F^{(i)}(v_h)$$

对于每一个时空区域的离散，我们有

- $\mathscr{T}_{N_i} = span\{\phi_l^i\}_l^{M_i}$, for $u_h^i \in S_h^p(\mathscr{T}_{N_i})$, 因此设 $u_h^i = \sum_{l=1}^{M_i} U_l^i \phi_l^i(x,t)$
- $S_h^p(\Sigma_h) = span\{\psi_n\}_{n=1}^{M_\Sigma}$, for $\lambda_h \in S_h^p(\Sigma_h)$, 因此设 $\lambda_h = \sum_{n=1}^{M_\Sigma} \lambda_\Sigma^n \psi_n(x,t)$

因此我们有线性形式

$$\begin{cases} A_{II}^{(i)}[k,\ell] = A^{(i)}\left(\varphi_\ell^i, \varphi_k^i\right) + C^{(i)}\left(\varphi_\ell^i, 0; \varphi_k^i, 0\right) & \text{for } k,\ell = 1,\ldots,M_i, \\ A_{I\Sigma}^{(i)}[k,n] := C^{(i)}\left(0, \psi_n; \varphi_k^i, 0\right) & \text{for } k = 1,\ldots,M_i \text{ and } n = 1,\ldots,M_\Sigma \\ A_{\Sigma I}^{(i)}[m,\ell] := C^{(i)}\left(\varphi_\ell^i, 0; 0, \psi_m\right) & \text{for } m = 1,\ldots,M_\Sigma \text{ and } \ell = 1,\ldots,M_i \\ A_{\Sigma\Sigma}[m,n] := \sum_{i=1}^{P} C^{(i)}\left(0, \psi_n; 0, \psi_m\right) & \text{for } m,n = 1,\ldots,M_\Sigma. \end{cases}$$

右端项为

$$F_I^{(i)}[k] := F^{(i)}\left(\varphi_k^i\right) \quad \text{for } k = 1,\ldots,M_i.$$

矩阵形式 可以写成矩阵形式：

$$\begin{pmatrix} A_{II}^{(1)} & & & & A_{I\Sigma}^{(1)} \\ & A_{II}^{(2)} & & & A_{I\Sigma}^{(2)} \\ & & \ddots & & \vdots \\ & & & A_{II}^{(P)} & A_{I\Sigma}^{(P)} \\ A_{\Sigma I}^{(1)} & A_{\Sigma I}^{(2)} & \cdots & A_{\Sigma I}^{(P)} & A_{\Sigma\Sigma} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_I^{(1)} \\ \boldsymbol{u}_I^{(2)} \\ \vdots \\ \boldsymbol{u}_I^{(P)} \\ \boldsymbol{\lambda}_\Sigma \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_I^{(1)} \\ \boldsymbol{f}_I^{(2)} \\ \vdots \\ \boldsymbol{f}_I^{(P)} \\ \mathbf{0} \end{pmatrix}.$$

可以发现这类矩阵是超大型稀疏矩阵，因为每个子矩阵都是稀疏的，且子矩阵也是稀疏的。因此如何避免求解这种超大型稀疏矩阵我们在第三章进行探讨。我们希望借助深度学习的力量来避免线性方程组的求解，以达到更快更准的目标。

三、DG 和 STDG 的数值实现

为了数值实现 DG 以及 STDG 方法，理论和实现其实并不完全一样。需要说明的是，目前实现 DG 方法大多以工业软件为主，或者是大型开源的 C++ 库。当然这些都是非常完善的软件，效率也一定非常高。但是由于接下来想要将其理论和深度学习方法融合，python 语言是离不开的。因此有必要用 python 再实现一些简单的数值算例。而 STDG 方法目前更是基本没有公开的代码。

下面介绍一些具体实施的细节。

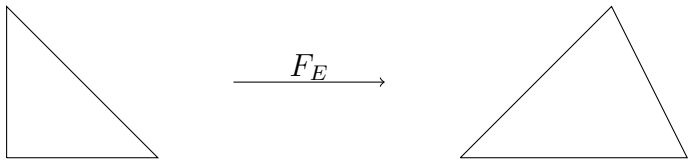
1 参考单元和真实单元

在实现 DG 方法的过程中，可以从理论上看出需要计算在单元上，内切面上的积分，例如三角形，线段等。在这里我们采取在参考单元上的坐标变换，只考虑在参考单元上的积分。

参考三角单元 我们将参考三角单元记作 $\hat{E}$，它的三个顶点分别为 $(0,0),(1,0),(0,1)$。对于任意一个三角单元来说 E, 我们有仿射变换 $F_E : \hat{E} \to E$. 假设 E 的三个顶点分别为 $(x_i, y_i), i = 1, 2, 3$, 那么我们有

$$\begin{pmatrix} x \\ y \end{pmatrix} = F_E \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = B_E \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} + b_E = \begin{pmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} + \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

简单示意图如下：



自然的我们有逆变换：

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = F_E^{-1} \begin{pmatrix} x \\ y \end{pmatrix} = B_E^{-1} \left(\begin{pmatrix} x \\ y \end{pmatrix} - b_E \right)$$

同时我们可以发现 B_E 的行列式的值在积分计算中有出现。若将 $|E|$ 记作单元 E 的面积, 那么我们有

$$det(B_E) = 2|B_E|$$

因此当我们有定义在参考单元 $\hat{E}$ 上的函数 $\hat{v}$ 时, 有

$$\hat{v} = v \circ F_E; \qquad v = \hat{v} \circ F_E^{-1}$$

因此 v 和 $\hat{v}$ 的梯度有如下关系:

$$\nabla\hat{v} = B_E^T \nabla v \circ F_E; \qquad \nabla v = B_E^{-T} \nabla\hat{v} \circ F_E^{-1}$$

2 基函数

在这里我们设局部基函数为阶数不超过 K 的多项式。例如

$$\hat{\phi}_i(\hat{x}, \hat{y}) = \hat{x}^i \hat{y}^j, i + j = k$$

那么基函数空间项数为 $N_{loc} = \frac{(k+1)(k+2)}{2}$.

具体来说我们有如下几种常用的基函数空间，以 2 维为例：

分段线性函数:

$$\hat{\phi}_0(\hat{x}, \hat{y}) = 1, \hat{\phi}_1(\hat{x}, \hat{y}) = \hat{x}, \hat{\phi}_2(\hat{x}, \hat{y}) = \hat{y}$$

分段二次函数:

$$\hat{\phi}_0(\hat{x}, \hat{y}) = 1, \hat{\phi}_1(\hat{x}, \hat{y}) = \hat{x}, \hat{\phi}_2(\hat{x}, \hat{y}) = \hat{y}, \hat{\phi}_3(\hat{x}, \hat{y}) = \hat{x}^2, \hat{\phi}_4(\hat{x}, \hat{y}) = \hat{x}\hat{y}, \hat{\phi}_5(\hat{x}, \hat{y}) = \hat{y}^2.$$

3 数值积分

由于映射 $F_E : \hat{E} \to E$ 为仿射变换, 并且我们有

$$\int_E v = \int_{\hat{E}} v \circ F_E \det(\boldsymbol{B}_E) = 2|E| \int_{\hat{E}} \hat{v}.$$

那么积分可以由如下方式逼近：

$$\int_E v \approx 2|E| \sum_{j=1}^{Q_{\mathrm{D}}} w_j \hat{v}\left(s_{x,j}, s_{y,j}\right).$$

若积分设计向量函数和梯度，则有

$$\begin{aligned}\int_E \nabla u \cdot \boldsymbol{v} &= 2|E| \int_{\hat{E}} \left(\boldsymbol{B}_E^T\right)^{-1} \nabla \hat{u} \cdot \hat{\boldsymbol{v}} \\ &\approx 2|E| \sum_{j=1}^{Q_{\mathrm{D}}} w_j \left(\boldsymbol{B}_E^T\right)^{-1} \nabla \hat{u}\left(s_{x,j}, s_{y,j}\right) \cdot \hat{\boldsymbol{v}}\left(s_{x,j}, s_{y,j}\right).\end{aligned}$$

类似的，如果积分关于 v,w 的梯度都有涉及，我们有

$$\int_E \nabla u \cdot \nabla v \approx 2|E| \sum_{j=1}^{Q_{\mathrm{D}}} v_j \left(\boldsymbol{B}_E^T\right)^{-1} \nabla \hat{u}\left(s_{x,j}, s_{y,j}\right) \cdot \left(\boldsymbol{B}_E^T\right)^{-1} \nabla \hat{v}\left(s_{x,j}, s_{y,j}\right).$$

4 DG 数值装配方案

4.1 一些细节

对于理论部分所述关于利用 DG 方法求解 Possion 方程部分，我们设计了对称系数以及惩罚因子 ϵ, σ_0，并且有如下结论：

- 若 $\epsilon = -1$, 方法变为对称型内罚伽辽金方法 (symmetric interior penalty Galerkin)。这种方法在惩罚因子 σ 足够大的时候收敛。
- 若 $\epsilon = +1$, 方法变为非对称型内罚伽辽金方法 (nonsymmetric interior penalty Galerkin). 方法对于任意非负惩罚因子 σ_e^0 都收敛。
- 若 $\epsilon = 0$, 方法变为缺型内罚伽辽金方法 (incomplete interior penalty Galerkin). 这种方法在惩罚因子 σ 足够大的时候收敛。
- J_1 是额外的稳定项。对于这一项中的惩罚因子 σ^1 通常设为 0。

首先我们将 $A_h(u,v)$ 分为两部分：

$$\begin{cases} T_1 = (a\nabla u, \nabla v)_{\mathcal{T}_h} \\ T_2 = -\left(\langle\{a\mathbf{n}\cdot\nabla u\}, [v]\rangle_{\mathcal{E}_h^{ID}} - \epsilon\langle\{a\mathbf{n}\cdot\nabla v\}, [u]\rangle_{\mathcal{E}_h^{ID}}\right) + \sum_{e\in\mathcal{E}_h^{ID}} \dfrac{\sigma_e^0}{|e|^{\beta_0}} \displaystyle\int_e [u]\,[v] \end{cases}$$

因此我们有

$$\forall 1 \leq i,j \leq N_{\text{loc}}, \quad (T_1)_{i,j} = \int_E (\nabla \phi_{j,E} \cdot \nabla \phi_{i,E})$$

施加坐标仿射变换 F_E, 我们能够将参考单元上的积分算出:

$$(T_1)_{i,j} = 2|E| \int_{\hat{E}} \left(\left(\boldsymbol{B}_E^T\right)^{-1} \hat{\nabla} \hat{\phi}_i \cdot \left(\boldsymbol{B}_E^T\right)^{-1} \hat{\nabla} \hat{\phi}_j \right).$$

右边项 $\boldsymbol{b}_E$ 对应的可以写为

$$(\boldsymbol{b}_E)_i = \int_E f \phi_{i,E} = det(B_E) \int_{\hat{E}} f \hat{\phi}_i.$$

现在我们可以计算对一个固定的内切面 e 上积分所对应的局部矩阵：

$$T_2 = -\left(\langle \{a\mathbf{n} \cdot \nabla u\}, [v] \rangle_{\mathcal{E}_h^{ID}} - \epsilon \langle \{a\mathbf{n} \cdot \nabla v\}, [u] \rangle_{\mathcal{E}_h^{ID}} \right) + \sum_{e \in \mathcal{E}_h^{ID}} \frac{\sigma_e^0}{|e|^{\beta_0}} \int_e [u]\,[v].$$

记 u 和 v 限制在单元 E_i 上的函数为 u_i 和 v_i。写成跳量形式并展开，我们可以得到：

若 e 为输入内部的内切面:

$$T_e = -\int_e \{\nabla u \cdot \boldsymbol{n}_e\}\,[v] + \epsilon \int_e \{\nabla v \cdot \boldsymbol{n}_e\}\,[u] + \frac{\sigma_e^0}{|e|^{\beta_0}} \int_e [u]\,[v].$$

那么其可以写为

$$T_e = m_e^{11} + m_e^{22} + m_e^{12} + m_e^{21}$$

其中

$$\begin{cases} m_e^{11} = -\dfrac{1}{2} \displaystyle\int_e \nabla u_{h,1} \cdot \boldsymbol{n}_e v_1 + \frac{\epsilon}{2} \int_e \nabla v_1 \cdot \boldsymbol{n}_e u_{h,1} + \frac{\sigma_e^0}{|e|^{\beta_0}} \int_e u_{h,1} v_1, \\ m_e^{22} = \dfrac{1}{2} \displaystyle\int_e \nabla u_{h,2} \cdot \boldsymbol{n}_e v_2 - \frac{\epsilon}{2} \int_e \nabla v_2 \cdot \boldsymbol{n}_e u_{h,2} + \frac{\sigma_e^0}{|e|^{\beta_0}} \int_e u_{h,2} v_2, \\ m_e^{12} = -\dfrac{1}{2} \displaystyle\int_e \nabla u_{h,2} \cdot \boldsymbol{n}_e v_1 - \frac{\epsilon}{2} \int_e \nabla v_1 \cdot \boldsymbol{n}_e u_{h,2} - \frac{\sigma_e^0}{|e|^{\beta_0}} \int_e u_{h,2} v_1, \\ m_e^{21} = -\dfrac{1}{2} \displaystyle\int_e \nabla u_{h,1} \cdot \boldsymbol{n}_e v_2 + \frac{\epsilon}{2} \int_e \nabla v_2 \cdot \boldsymbol{n}_e u_{h,1} - \frac{\sigma_e^0}{|e|^{\beta_0}} \int_e u_{h,1} v_2. \end{cases}$$

这四项都是大小为 $N_{loc} \times N_{loc}$ 的二维局部矩阵, 即可以记为 $M_e^{11}, M_e^{22}, M_e^{12}, M_e^{21}$, 因此我们将基函数分别代入可以得到:

$$\begin{cases} \left(\boldsymbol{M}_e^{11}\right)_{ij} = -\dfrac{1}{2}\displaystyle\int_e \nabla\phi_{j,E_e^1} \cdot \boldsymbol{n}_e \phi_{i,E_e^1} + \dfrac{\epsilon}{2}\int_e \nabla\phi_{i,E_e^1} \cdot \boldsymbol{n}_e \phi_{j,E_e^1} + \dfrac{\sigma_e^0}{|e|^{\beta_0}}\displaystyle\int_e \phi_{j,E_e^1}\phi_{i,E_e^1} \\ \left(\boldsymbol{M}_e^{22}\right)_{ij} = \dfrac{1}{2}\displaystyle\int_e \nabla\phi_{j,E_e^2} \cdot \boldsymbol{n}_e \phi_{i,E_e^2} - \dfrac{\epsilon}{2}\int_e \nabla\phi_{i,E_e^2} \cdot \boldsymbol{n}_e \phi_{j,E_e^2} + \dfrac{\sigma_e^0}{|e|^{\beta_0}}\displaystyle\int_e \phi_{j,E_e^2}\phi_{i,E_e^2} \\ \left(\boldsymbol{M}_e^{12}\right)_{ij} = -\dfrac{1}{2}\displaystyle\int_e \nabla\phi_{j,E_e^2} \cdot \boldsymbol{n}_e \phi_{i,E_e^1} - \dfrac{\epsilon}{2}\int_e \nabla\phi_{i,E_e^1} \cdot \boldsymbol{n}_e \phi_{j,E_e^2} - \dfrac{\sigma_e^0}{|e|^{\beta_0}}\displaystyle\int_e \phi_{j,E_e^2}\phi_{i,E_e^1} \\ \left(\boldsymbol{M}_e^{21}\right)_{ij} = \dfrac{1}{2}\displaystyle\int_e \nabla\phi_{j,E_e^1} \cdot \boldsymbol{n}_e \phi_{i,E_e^2} + \dfrac{\epsilon}{2}\int_e \nabla\phi_{i,E_e^2} \cdot \boldsymbol{n}_e \phi_{j,E_e^1} - \dfrac{\sigma_e^0}{|e|^{\beta_0}}\displaystyle\int_e \phi_{j,E_e^1}\phi_{i,E_e^2} \end{cases}$$

若 e 为属于边界上的内切面, 那么我们只有

$$\left(\boldsymbol{M}_e^{11}\right)_{ij} = -\int_e \nabla\phi_{j,E_e^1} \cdot \boldsymbol{n}_e \phi_{i,E_e^1} + \epsilon\int_e \nabla\phi_{i,E_e^1} \cdot \boldsymbol{n}_e \phi_{j,E_e^1} + \frac{\sigma_e^0}{|e|^{\beta_0}}\int_e \phi_{j,E_e^1}\phi_{i,E_e^1}$$

并且局部的右端项 b_e 为

$$(b_e)_i = \epsilon\int_e \left(\nabla\phi_{i,E_e^1} \cdot \boldsymbol{n}_e + \frac{\sigma_e^0}{|e|^{\beta_0}}\phi_{i,E_e^1}\right) g_D$$

在这里我们设 $\beta_0 = 1, \sigma_0 = 10 * p^2, \epsilon = -1$

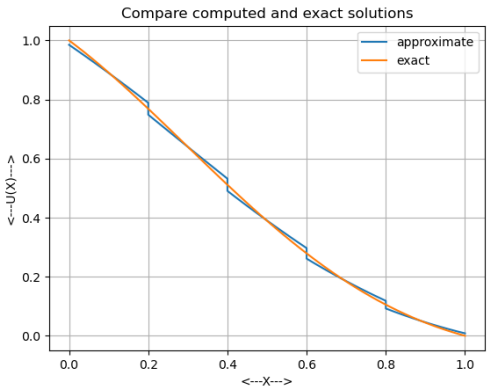
4.2 数值结果

编写语言:Python 3.11

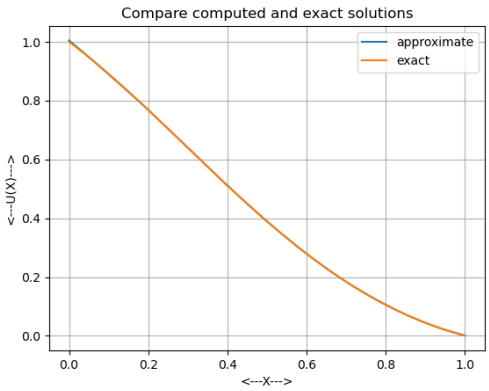
1D 问题

我们假设精确的解析解为 $u(x) = (1-x)e^{-x^2}$.

下面分别是利用 DG 方法求出的数值解和精确解的图像

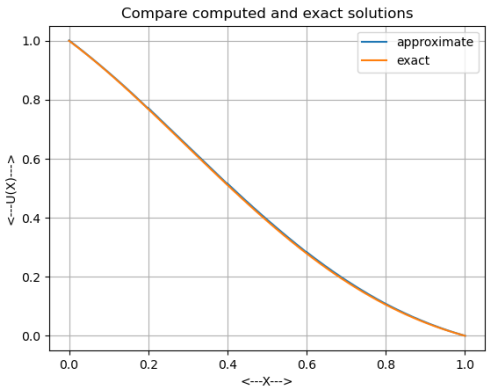


(a) 5 verticles with SIPG $\sigma = 1$

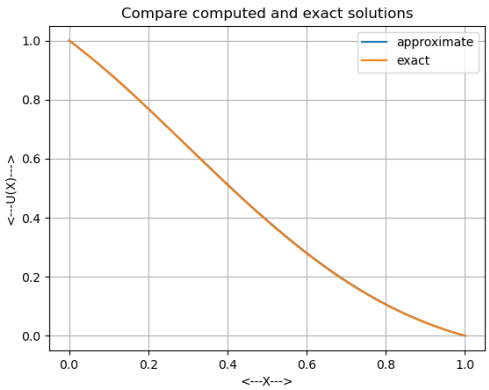


(b) 5 verticles with SIPG $\sigma = 5$

图 4.1: SIPG



(a) 5 verticles with NIPG $\sigma = 1$



(b) 5 verticles with NIPG $\sigma = 5$

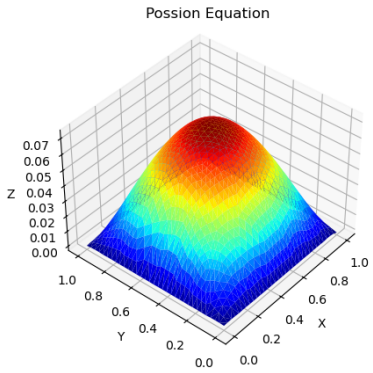
图 4.2: NIPG

2D 问题

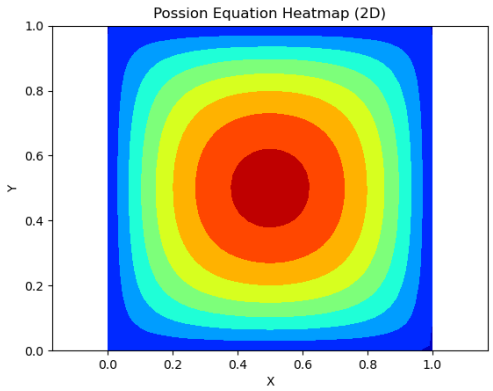
在这里我们使用了 python 中 Triangle 库来生成 2D 网格, 具体的参数设置见附录。我们可以通过设置每个单元，三角形，的最大的面积以及生成的最大角度来控制我们想要的三角划分，同时它提供了生成不规则区域的边界方法。

—-齐次狄利克雷边界—-

下面是有齐次边界条件的 Possion 方程解的 3D 以及热力示意图, 其中 $f = 1$

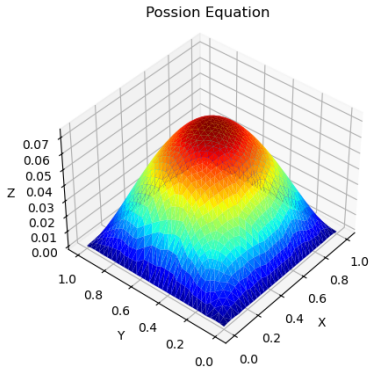


(a) SIPG $\sigma = 10$

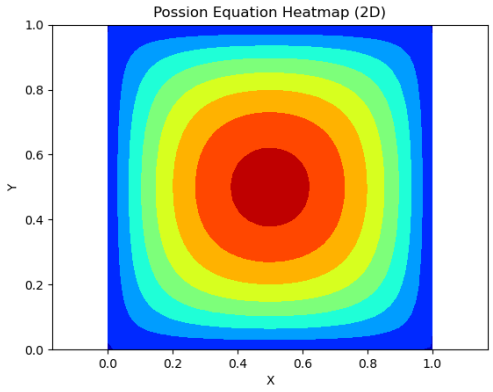


(b) SIPG $\sigma = 10$

图 4.3: SIPG

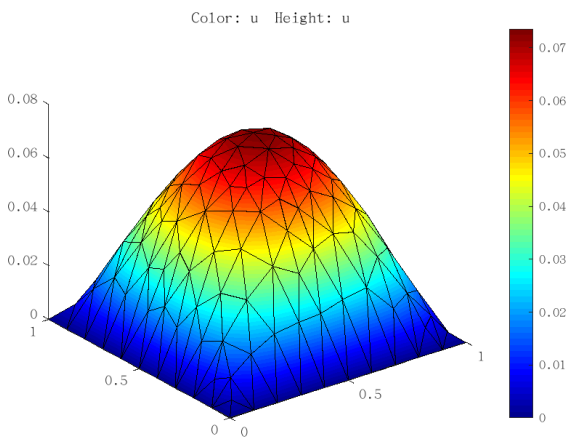


(a) NIPG $\sigma = 10$

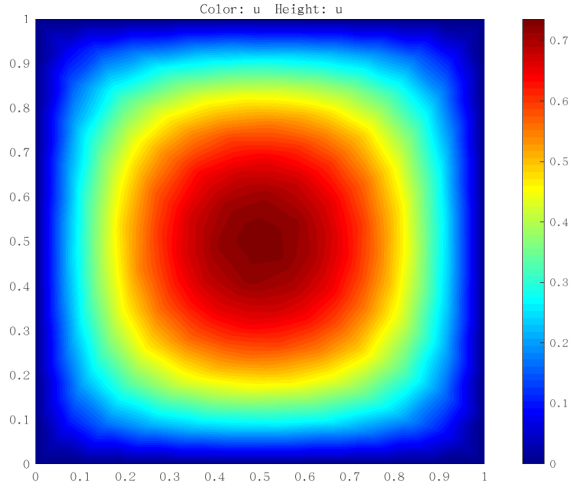


(b) NIPG $\sigma = 10$

图 4.4: NIPG



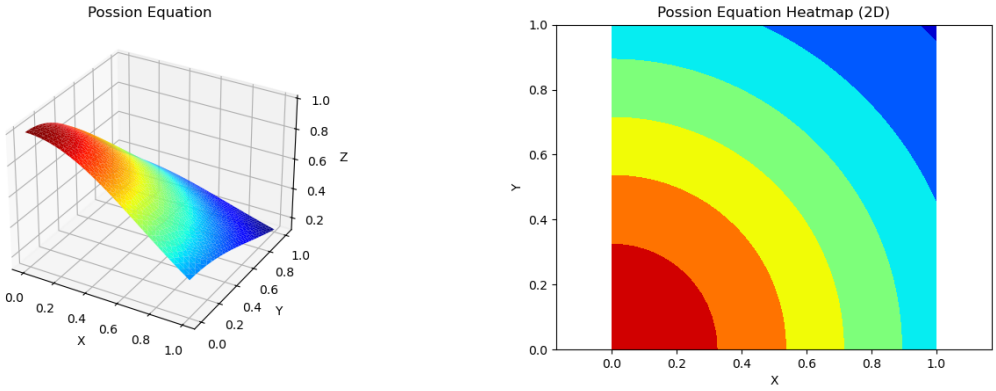
(a) 3D plot by matlab



(b) heat map by matlab

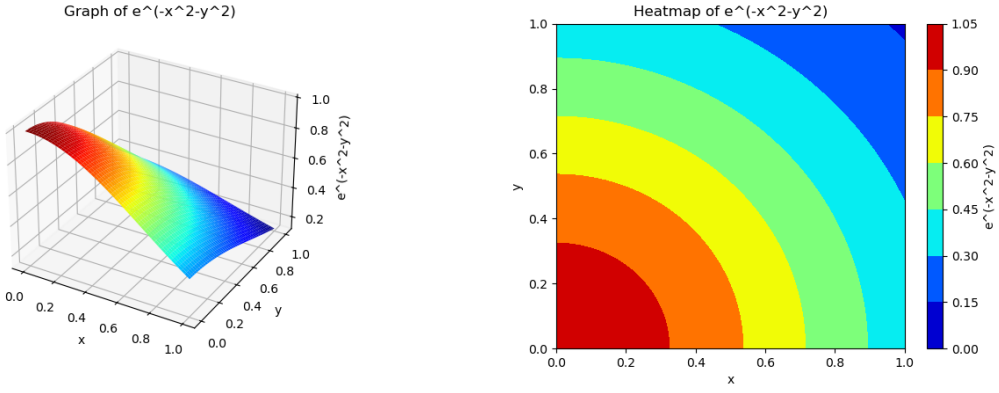
—-$u(x,y) = e^{-x^2-y^2}$—-

解析解为 $u(x,y) = e^{-x^2-y^2}$，主要考察程序在非其次边界条件是否成立。



(a) 3Dplot **(b) heatmap**

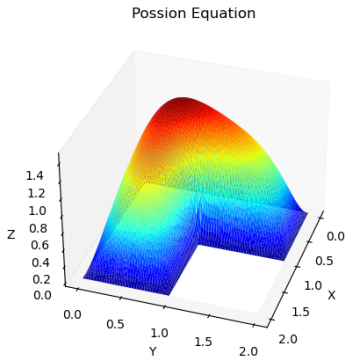
图 4.6: SIPG



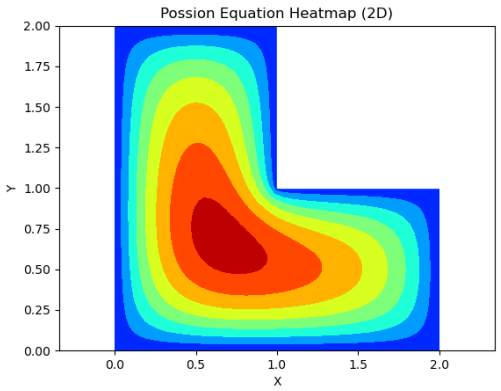
(a) exact 3D **(b) Exact heat**

—-不规则边界—-

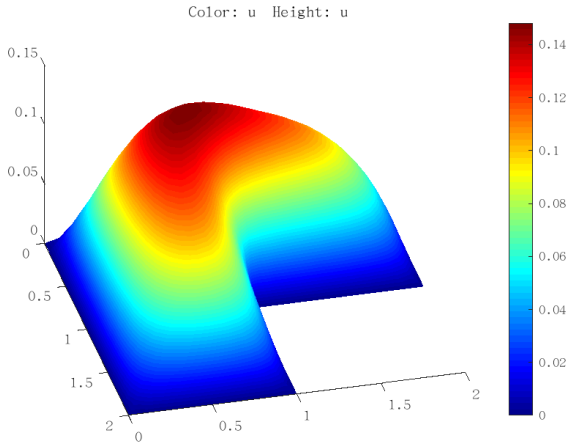
. $f = 10$ with homogeneous dirichlet boundary condition.



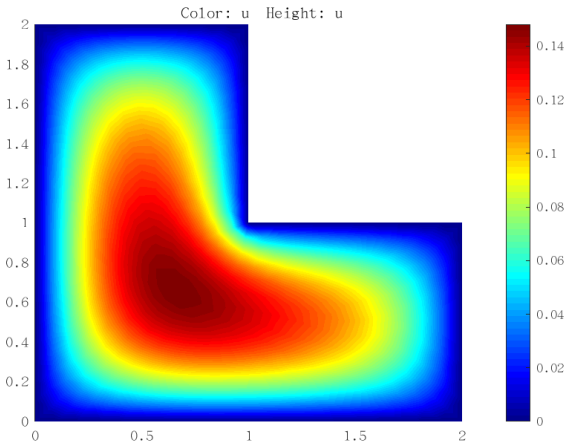
(a) 3Dplot



(b) heatmap



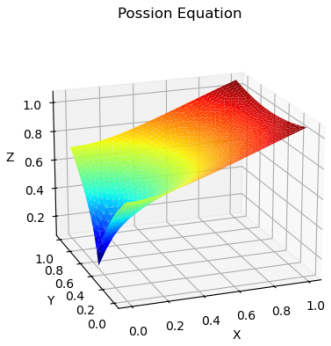
(c) matlab 3D



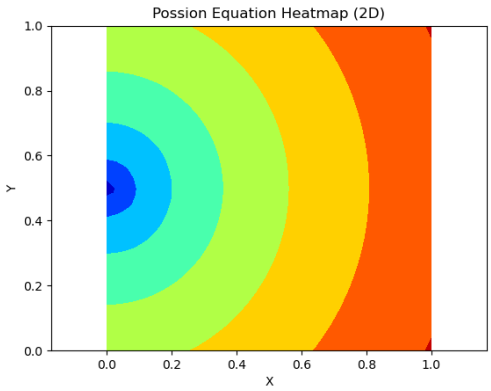
(d) matlab heat

—-有奇点情况—-

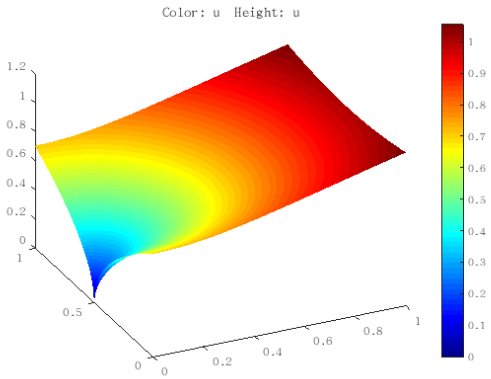
. 解析解为 $(x^2 + (y - \frac{1}{2})^2)^{\frac{1}{4}}$



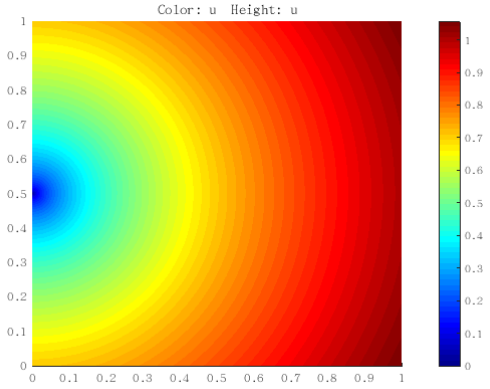
(a) 3Dplot



(b) heatmap



(c) matlab 3D



(d) matlab heat

5 时空间断加辽金方法 (STDG) 方案

5.1 一些细节

利用时空间断伽辽金方法处理 1D 问题和上述处理 2Dpossion 方程有一定的相似之处，但由于原理上有些差异，我们也叙述其求解细节，将其数值实现。由 4-2可见，我们当然可以以相同的方法得到 A_E 和 M_{ij}. 除此之外，由 (4-3) 可以得到:

若 e 是属于内部的内切面，并且由于迎风量决定我们有当 $n_{E_e^1,t} > 0$，

$$\begin{cases} (N_e^{11})_{ij} = \int_e \phi_{j,E_e^1} \phi_{i,E_e^1} \cdot n_{E_e^1,t} \\ (N_e^{21})_{ij} = -\int_e \phi_{j,E_e^1} \phi_{i,E_e^2} \cdot n_{E_e^1,t} \end{cases}$$

否则 $n_{E_e^1,t}$

$$\begin{cases}(N_e^{12})_{ij} = \int_e \phi_{j,E_e^2}\phi_{i,E_e^1} \cdot n_{E_e^1,t} \\ (N_e^{22})_{ij} = -\int_e \phi_{j,E_e^2}\phi_{i,E_e^2} \cdot n_{E_e^1,t}\end{cases}$$

若 e 是 Σ_T 上的边, 那么可以知道在正方形区域上 $n_{E_e,t}$ 永远是 1，因此有:

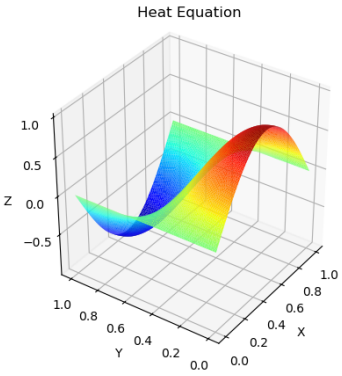
$$(N_e)_{ij} = \int_e \phi_{j,E_e}\phi_{i,E_e} \cdot n_{E_e,t}$$

在这里值的注意的是我们将涉及 Σ_0 上的边上积分移到了右边, 并且将 (4-3) 的第一项添加到了上面的 A_E 中。

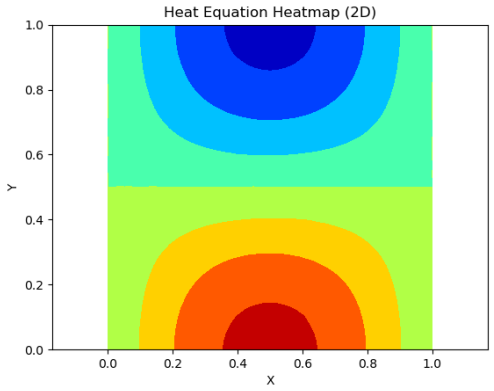
5.2 数值结果

5.2.1 Regular Solution

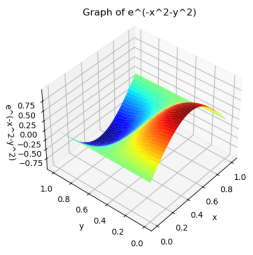
解析解为 $u(x,t) = cos(\pi t)sin(\pi x)$



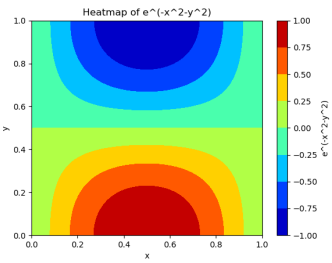
(a) 3Dplot



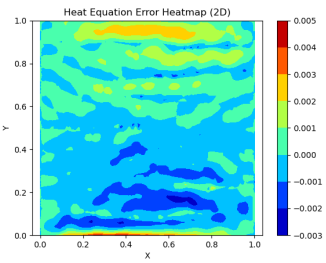
(b) heatmap



(c) matlab 3D

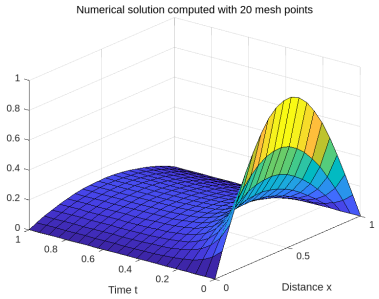


(d) matlab heat

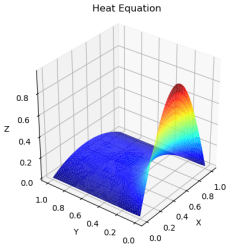


(e) 和解析解误差，网格大小 0.01

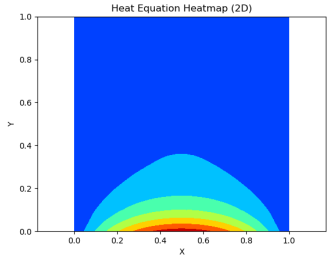
$f = 1$ 初始条件为 $sin(\pi x)$



(a) Matlab



(b) ST Method 3D



(c) ST Method Heat

四、深度学习架构

随着机器学习技术的不断发展，深度学习已经在 PDE 数值仿真领域取得了一系列的进展，包括 PDE 正反问题的仿真，物理场重建和模拟等等。他们或是利用优化或是利用数据的形式，针对一些特定的问题，例如高维问题，流体问题上提出了很多积极的方案，大大加快了 AI 在 Science 领域的发展。这些方法往往能够避免一些不必要的离散误差，并且能够结合数据中的信息。下面简单叙述当今深度学习在 PDE 领域的几个框架。

1 物理信息神经网络

基于深度学习的 PDE 求解的框架分类其实有很多种，在这里按照深度学习原理大致可以分为三种。

第一种是基于机理驱动的方式学习 PDE；在此框架下主要衍生出以 PDE 的强形式和弱形式进行学习的两种方式。关于强形式学习方案，最早的工作是 Raissi 在 2017 年提出的物理信息神经网络 (PINN)[4]，利用神经网络作为代理模型，它将方程以强形式加入到损失函数中进行网络约束。在此之后基于平凡的 PINN 的方法衍生出很多针对不同问题的方法，例如针对不规则区域问题的 PhyGeoNet[5]，针对流体问题的 NSFnets[6]，针对分数阶方程的 fPINN 等等。而挂于方程弱形式的学习方案目前的工作较少，因为涉及积分项的运算。这些方案主要以优化方案为主，数据往往以边界条件的方式给出。因此对于一些有多尺度形式的解来说，例如高频振荡解，模型有时候很难学习成功。

第二种是基于数据驱动的方式学习 PDE；在此框架下主要以神经算子学习方式为主。主要思想是利用大量的数据学习函数空间到函数空间的映射，大大提高了模型的泛化能力，并且模型和网格离散化无关。经典的工作有 DeepONet[7] 和 FNO[8]，这两种方法都已经在多个领域发挥了强有力的作用，例如天气预测，地质勘探等。

第三种是数据和机理相结合的方式学习 PDE；数据和机理方程相符相称，该方式在正反问题上都有积极的作用，且在机理驱动的方案中，方程的边界条件常常以数据的方式给出。

下面主要研究弱形式学习方案。

2 弱形式网络

弱解形式在数学和物理问题中有着重要的作用，特别是在偏微分方程和变分问题中。尤其是对于某些情况下，问题的解可能存在不光滑甚至存在奇点的情况。同时在数值计算值中，弱解形式更有利于数值稳定性和收敛性。

然而针对于弱解形式的神经网络算法目前很少。在 2017 年 Weinan E 最早提出针对波动类方程提出以能量积分最小化的形式 (Deep Ritz Method)[9]进行约束，将 PDE 以积分形式引入到神经网络中。随后 Zang 在 2020 年提出一种针对弱解形式的深度学习方案 (Weak Adervaserial Network)[10]。

在传统的数值方法中，处理弱解形式的主要数值方法集中在有限体积以及有限元类方法，这些方法中很大一部分实施困难在与基函数的选取以及装配刚度矩阵。即使刚度矩阵装配完成，方程组的求解也面临着严重的稀疏性灾难。

当前有些一些工作涉及到有限元方法和深度学习方法结合，例如 FEA-Net[11]，但是可以看到工作的重点还是集中在处理刚度矩阵上，本质上并没有避免基函数选取导致的一些积分问题。下面介绍经典的处理弱解形式的深度学习方案。

2.1 弱对抗神经网络

弱对抗神经网络 (Weak Adervaserial Network) 首先提出了处理弱形式问题的方法。该工作提出在弱解形式下，用单个神经网络作为代理模型，在写为弱形式之后，测试函数同样选取在神经网络空间，最后形成对抗训练的形式进行优化求解。

以对流扩散方程为例，方程强形式可以写为：

$$\nabla \cdot (a\nabla u)u + b \cdot \nabla u + cu = f$$

其中 a 为矩阵，b 为向量

方程弱形式可以写为:

$$\begin{cases} \langle A[u], \phi \rangle = \int_\Omega (a\nabla u \nabla \phi + b\nabla u\phi + cu\phi - f\phi) = 0 \quad x \in \Omega \\ u = g \quad x \in \partial\Omega \end{cases}$$

其中测试函数 $\phi \in H_0^1$。因此可以定义范数

$$\|A[u]\| = \max\{\langle A[u], \phi\rangle / \|\phi\|_2 \,|\phi \in H_0^1, \phi \neq 0\}$$

其中

$$||\phi||_2 = \left(\int_\Omega |\phi(x)|^2 dx\right)^{\frac{1}{2}}$$

因此可以发现 u 为该对流扩散方程的弱解当仅当 $||A[u]|| = 0$ 且边界上满足条件。因此问题等价转化成

$$\min_{u\in H^1} \|A[u]\|^2 \Leftrightarrow \min_{u\in H^1} \max_{\phi\in H_0^1}\{\langle A[u], \phi\rangle / ||\phi||_2\}$$

这种 min-max 的形式是标准的对抗神经网络的优化形式，因此可以形成测试函数和代理模型的对抗训练。同时随着数据的加入可以实现反问题，如方程中的系数 a, b 的预测。

2.2 ParticleWNN

在 2023 年同样是 Zang 提出了 ParticleWNN 的工作[12]，同样是针对弱解形式的方程。在这篇工作中，函数空间类似的取在神经网络函数空间，但是不同的是测试函数选取在紧支集为圆的函数空间上，这样避免了求积分时的边界上的积分。

下面以 Possion 方程为例介绍该工作。

2.2.1 框架

首先拥有 Dirichlet 边界的 Possion 方程可以写成如下形式:

$$\begin{cases} -\Delta u = f \quad x \in \Omega \\ u = g \quad x \in \partial\Omega \end{cases}$$

Possion 方程的弱解形式为：求 $\{u \in H^1(\Omega), u|_{\partial\Omega} = g\}$ s.t.

$$\int_\Omega \nabla u \cdot \nabla \phi dx = \int_\Omega f\phi dx$$

其中 $\phi \in H_0^1$，因此自然不含有 Ω 上的边界积分。

设区域上方程的代理模型为一个多层神经网络，例如含有三个隐藏层，每层中含有 50 个神经元。接受的输入为 (x, y), 输出为 $u(x, y)$。因此方程弱解残差形式可以写为

$$R(u_{NN}, \phi) = \int_{\Omega} \nabla u_{NN} \cdot \nabla \phi dx - \int_{\Omega} f \phi dx$$

和 WAN 不同的是，在这里测试函数经过了仔细的设置：取了紧支径基函数 (CSRBF)。通常定义在半径大小为 r 的圆上，有如下形式：

$$\phi(r) = \begin{cases} \phi_+(r) & r(x) \leq 1 \\ 0 & r(x) > 1 \end{cases}$$

具体来说可以设置为 Wendland's CSRBFs:

$$\phi_{d,2}(r) = \begin{cases} \frac{(1-r)^{l+2}}{3} \left[(l^2+4l+3)r^2 + (3l+6)r + 3\right] & r(x) \leq 1 \\ 0 & r(x) > 1 \end{cases}$$

其中 $l = [d/2] + 3, [\cdot]$ 为向下取整。一维定义在参考单元上的图像如下：

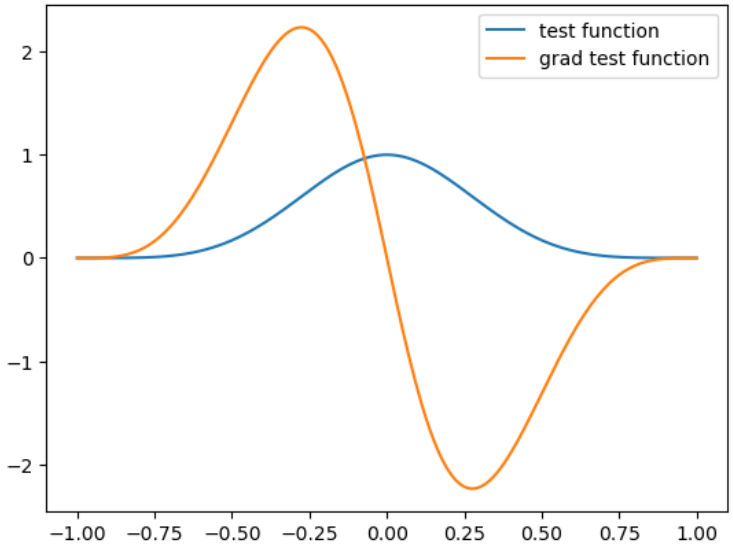


图 2.1: Wendland's CSRBFs

可以发现函数以及导数图像在边界处为 0。

2.2.2 约束组成 (损失函数)

参考单元的坐标记为 $\boldsymbol{s}$，那么真实的物理区域有坐标仿射变换 $\boldsymbol{x} = \boldsymbol{s}R_i + \boldsymbol{x}_i^c$, 将其代入上式残差形式中有:

$$\begin{aligned}\mathcal{R}\left(u_{NN};\varphi_i\right) &= \int_{B\left(\boldsymbol{x}_i^c,R_i\right)} \nabla_{\boldsymbol{x}} u_{NN}\cdot\nabla_r\varphi_i\cdot\nabla_{\boldsymbol{x}} r d\boldsymbol{x} - \int_{B(\boldsymbol{x}^c,R_i)} f\varphi_i d\boldsymbol{x},\\ &= R_i^d\left(\int_{B(\mathbf{0},1)} \nabla_{\boldsymbol{x}} u_{NN}(\boldsymbol{x})\cdot\nabla_r\varphi_i\cdot\nabla_{\boldsymbol{x}} r d\boldsymbol{s} - \int_{B(\mathbf{0},1)} f(\boldsymbol{x})\varphi_i d\boldsymbol{s}\right),\end{aligned}$$

其中由链式法则 $\nabla_{\boldsymbol{x}}\varphi_i = \nabla_r\varphi_i\cdot\nabla_{\boldsymbol{x}} r$. 文中在 $B(\mathbf{0},1)$ 中随机选取了 Nint 个积分点，对 N_p 个积分区域进行数值积分，也即对 N_p 个小圆上积分。假设 $\{\boldsymbol{s}_k\}_{k=1}^m$ 为 K_{int} 个积分点，且 $\{w_k\}_{k=1}^m$ 为对应的积分权重。那么将 $\boldsymbol{x}_k^{(i)}$ 记作 $\boldsymbol{s}_k * R_i + \boldsymbol{x}_i^c$。

则代理模型在单个圆上的损失函数为 $\mathcal{R}\left(u_{NN};\phi_i\right)$ 可以写作：

$$\mathcal{R}\left(u_{NN};\varphi_i\right) \approx \frac{R_i^d V_{B(\mathbf{0},1)}}{K_{\text{int}}}\sum_{k=1}^{K_{\text{int}}} w_k\left(\nabla_{\boldsymbol{x}} u_{NN}\left(\boldsymbol{x}_k^{(i)}\right)\cdot\nabla_r\varphi_i\left(\boldsymbol{x}_k^{(i)}\right)\cdot\nabla_{\boldsymbol{x}} r\left(\boldsymbol{x}_k^{(i)}\right) - f\left(\boldsymbol{x}_k^{(i)}\right)\varphi_i\left(\boldsymbol{x}_k^{(i)}\right)\right),$$

若共取了 N_p 个圆进行 particle loss 的计算，则有

$$\mathcal{L}_{\mathcal{R}} \approx \frac{\mathcal{V}_{B(\mathbf{0},1)}^2}{N_p K_{int}^2}\sum_{i=1}^{N_p} R_i^{2d}\left(\sum_{k=1}^{K_{\text{int}}} w_k\left(\nabla_{\boldsymbol{x}} u_{NN}\left(\boldsymbol{x}_k^{(i)}\right)\cdot\nabla_r\varphi_i\left(\boldsymbol{x}_k^{(i)}\right)\cdot\nabla_{\boldsymbol{x}} r\left(\boldsymbol{x}_k^{(i)}\right) - f\left(\boldsymbol{x}_k^{(i)}\right)\varphi_i\left(\boldsymbol{x}_k^{(i)}\right)\right)\right)^2$$

同时在边界上有约束

$$B = \sum_{j=1}^{N_{bd}}\left(u_{NN}(x_j) - g(x_j)\right)^2$$

因此最后损失函数可以设置为

$$\mathcal{L} = \mathcal{L}_{\mathcal{R}} + \mathcal{B}$$

3 Split Weak Net

在这里，我们受到 DG 方法中基函数选取的启发。由于基函数是定义在离散域上分片多项式函数，且数值解作为基函数的线性叠加。由广义逼近定理可以知道：具有至少一个隐藏层的神经网络，在理论上可以以任意精度逼近任何连续函数。具体来说，对于一个足够大的隐藏层，神经网络可以以任意精度逼近任何连续函数。

因此想到在分片区域上利用局部小网络来替代原本局部线性方程组求解的过程；也就是说原来由一个整体的神经网络作为代理模型变为由多个小网络组成的代理模型。这样的做的优势我认为主要有以下几点：

- 参数量由原来的多层神经网络参数 Θ 变为单层网络参数 $N_\theta \times N_{element}$
- 对于不规则边界，多尺度问题可以很自由的进行区域离散，分解操作
- 对于有间断解的问题允许交界面上的不连续
- 可以进行区域分解利用并行加速计算效率

在这里主要提出两种方案。

3.1 Split Particle WNN

该方法还是利用 Particle WNN 的框架和思路，但是在这里，我们并不采取随机策略选取 N_p 个积分单元，而是根据区域离散 (三角) 化的结果进行选取。例如将区域 Ω 离散化为 N_{elt} 个单元，则在单元中心作为圆心，半径不超过单元内切圆半径的长度作积分单元。同理可以在每个积分单元上通过放射变换定义测试函数，这样依旧避免了边界上的积分。

设置每个单元上的小网络选择方案有两种，一种是为仅含有少量例如 10 个神经元的单隐藏层，激活函数为 gelu 的神经网络，第二种是在输入之后进行一个特征变换，例如输入为 (x, y), 那么第一层的特征变换首先变为 (x, y, x^2, y^2, xy)。这里的特征变换和 DeepONet 中的 Trunk Net 类似，因此对于不同的问题可以选取不同的特征变换操作，例如对于涉及周期函数可以变为 $\cos(x), \sin(x), \cos(y), \sin(x), \ldots$，其实在这里设置特征变换的主要作用是为了增加网络拟合的非线性性，同时可以添加一些关于方程的先验知识。

接下来的 Loss 设置和 Particle WNN 类似，其中

$$\mathcal{L}_{\mathcal{R}} \approx \frac{\mathcal{V}^2_{B(\mathbf{0},1)}}{N_{elt} K^2_{int}} \sum_{i=1}^{N_{elt}} R_i^{2d} \left(\sum_{k=1}^{K_{\text{int}}} w_k \left(\nabla_{\boldsymbol{x}} u_{net_i} \left(\boldsymbol{x}_k^{(i)} \right) \cdot \nabla_r \varphi_i \left(\boldsymbol{x}_k^{(i)} \right) \cdot \nabla_{\boldsymbol{x}} r \left(\boldsymbol{x}_k^{(i)} \right) - f \left(\boldsymbol{x}_k^{(i)} \right) \varphi_i \left(\boldsymbol{x}_k^{(i)} \right) \right) \right)^2$$

边界约束项为

$$B(u) = \sum_{e \in \mathscr{E}^D} \int_e (u_{net} - g)^2$$

我们记区域一共被分解成了 $Nelt$ 个单元，在每个单元上定义网络 net_i 但是，为了约束最终解的光滑性，我们添加惩罚项

$$J_0 = \sum_{e \in \mathscr{E}^I} \int_e [u]^2; \qquad J_1 = \sum_{e \in \mathscr{E}^I} \int_e [\nabla u \cdot \mathbf{n}]$$

因此损失函数可以写为

$$\mathcal{L} = \mathcal{L}_{\mathcal{R}} + B + J_0 + J_1$$

数值结果可行性见后面一章。

注：

其实可以发现这里的 Loss 函数其实有四项，这里其实加大了网络优化的难度。但我认为分解成小网络区域，其实是解决区域之间耦合的方式，因此完全可以在解平滑的区域用粗网格离散，用大网络代理；在局部需要加密的地方用小网络进行替代，因此有时候优化难度大和离散化的方式也有很大关系。

3.2 Split Discontinuous Galerkin Net

其实可以看到 Split Particle WNN 沿用了 Particle WNN 测试函数的选取策略，在紧支集为圆的函数空间上选取，这是一个很好的选择，避免了边界上不必要的积分。

然而在 PWNN 的文章中也提到，这种测试函数的选取对于不同的问题来说可能差异非常大，并且很大程度上影响着最后数值结果的精度，也就是说在某种程度上并不能够通过测试函数的选取方式进行精度的控制。

对于一些拥有振荡解的问题来说，PWNN 往往通过增加随机选点的个数以及积分点的个数的方式来增加精度，这进一步增加了计算的复杂度。

考虑到在变分问题中，有限元类方法是非常强有力的工具。而对于有限元类方法三角划分是区域离散必不可少的一步，因此主要提出三种改进方案：

第一种解决方案是是否可以寻找紧支集为三角形的函数空间，替代紧支集为圆形的函数空间，从而不必随机选点，而是根据三角划分对于更需要的细化的地方进行加密取点。但这样的函数空间并不容易寻找。

第二种方案抛弃测试函数为单个的想法，接受边界上积分的麻烦。受到 DG 方法的启发，我们可以在单元上设置简单的基函数，例如设测试函数为分片多项式函数，则边界上的积分变得不可避免，但这对于间断有限元来说，并不困难。

第三种方案其实可以采取弱对抗神经网络中的策略，将测试函数取在神经网络函数空间中。但是由于是在每个小单元上取代理模型，因此测试函数数量会变大，训练会变得非

常困难。

因此采取第二种方案。首先为和经典 DG 方法一样，将测试函数设分片多项式函数，例如 2 阶多项式，则 Possion 方程的弱解形式变为

$$-\int_{\Omega}\Delta uv=\int_{\Omega}fv$$

离散化之后可以写作

$$-\sum_{E\in\Omega}\int_{E}\Delta uv=\sum_{E\in\Omega}\int_{E}fv$$

通过分部积分，对于单个单元 E 来说有：

$$\int_{E}\nabla u\cdot\nabla v-\int_{\partial E}(\nabla u\cdot\mathbf{n})v=\int_{E}fv$$

因此

$$\begin{aligned}\mathcal{L}_{dg}=&\sum_{i=1}^{Nloc}\sum_{j=1}^{Nelt}\left(\int_{E_j}\nabla u_j\cdot\nabla\phi_i-\int_{\partial E_j}(\nabla u_j\cdot\mathbf{n})\phi_i-\int_{E_j}f\phi_i\right)^2\\=&\sum_{i=1}^{Nloc}\sum_{j=1}^{Nelt}\left(\sum_{k=0}^{NintE}(w_k\nabla u_j\cdot\nabla\phi_i-f\phi_i)-\sum_{k=0}^{Ninte}w_k'(\nabla u_j\cdot\mathbf{n})\phi_i\right)^2\end{aligned}$$

其中 Nloc 为测试函数个数，NintE 为区域上积分点个数，Kinte 为内切面上积分点个数。

同理有边界约束项：

$$B(u)=\sum_{e\in\mathscr{E}^D}\int_{e}(u_{net}-g)^2$$

惩罚项

$$J_0=\sum_{e\in\mathscr{E}^I}\int_{e}[u]^2;\qquad J_1=\sum_{e\in\mathscr{E}^I}\int_{e}[\nabla u\cdot\mathbf{n}]$$

因此损失函数可以写为

$$\mathcal{L}=\mathcal{L}_{dg}+\mathcal{B}+J_0+J_1$$

和之前的间断有限元方法相比，避免了复杂的稀疏矩阵计算，相反通过直接学习局部函数形式来代替求解基函数的系数。理论上来说我们可以通过控制测试函数个数 $Nloc$ 来限制数值解的精度。

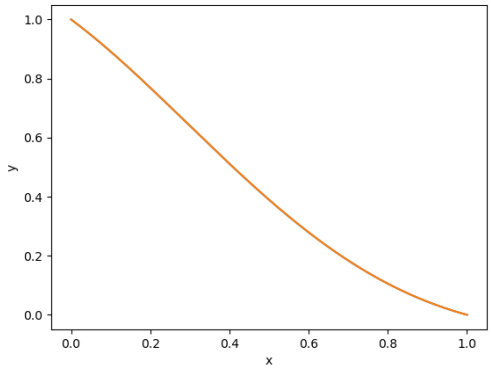
五、数值结果

下面是进行的数值结果，主要以 Particle WNN, Split Particle WNN 和 Split DGNet 为主。主要验证该方法的可行性，由于代码并未足够优化，因此暂时并没有完成训练速度的对比。

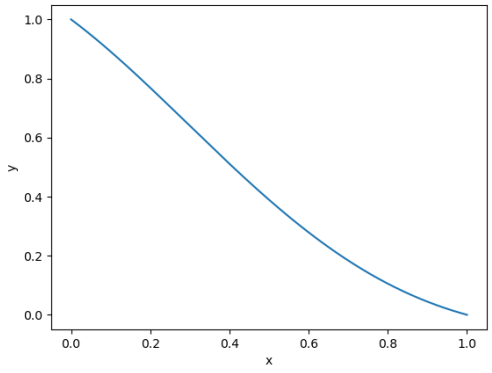
1 1D Possion

$$y = (1-x)e^{-x^2}$$

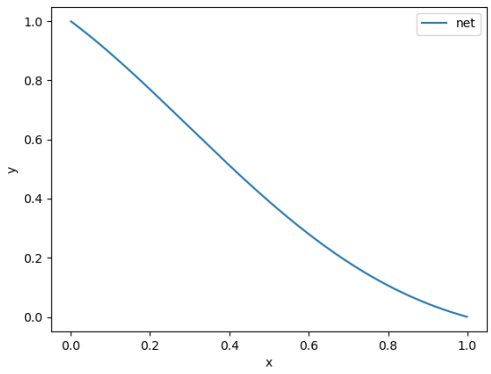
有解析解的情况：在这里特征变化为 (x, x^2), 只有一层隐藏层，无激活函数情况。离散化 h=0.1,Nint=50。



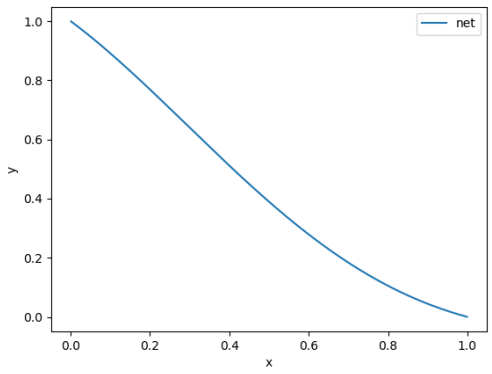
(a) exact solution



(b) Particle WNN



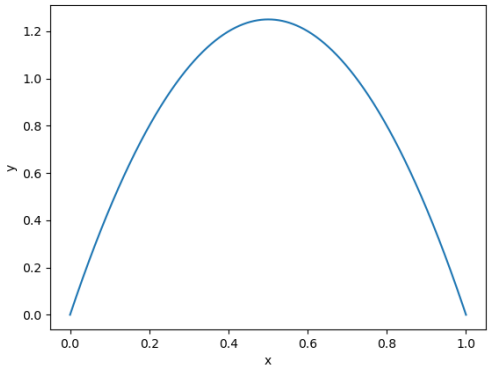
(c) Split Particle WNN



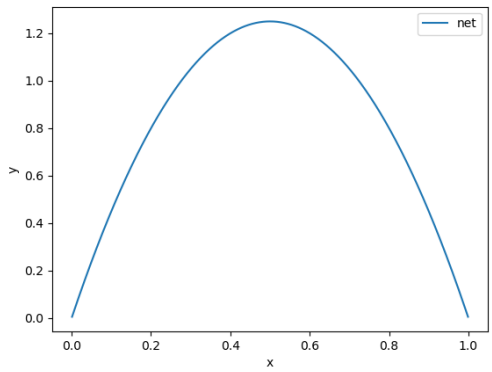
(d) Split DG Net

$f = 10$ with homogeneous DBC

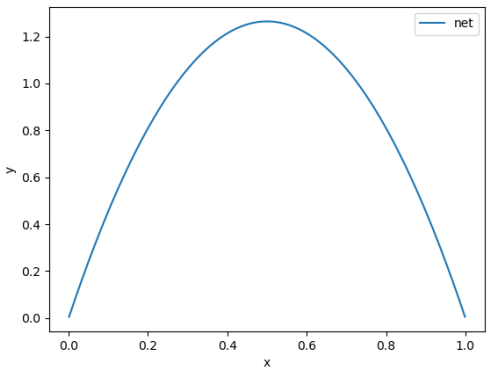
具有其次边界条件的 Possion 方程，网络参数和上述相同。



(a) Particle WNN



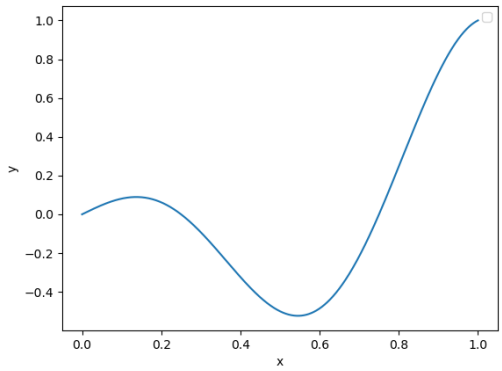
(b) Split Particle WNN



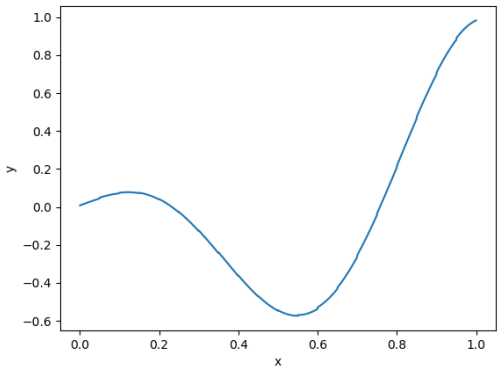
(c) Split DG Net

$y = xcos(\omega x)$

当 $w = 2\pi$ 时, 低频解：

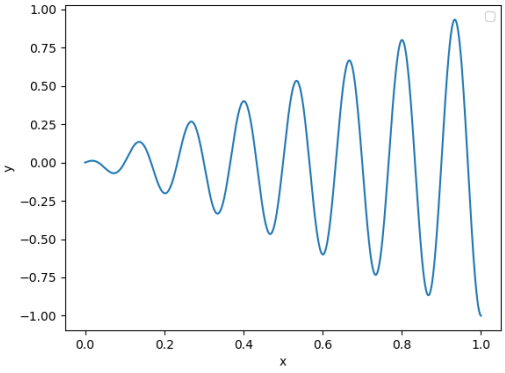


(a) Exact

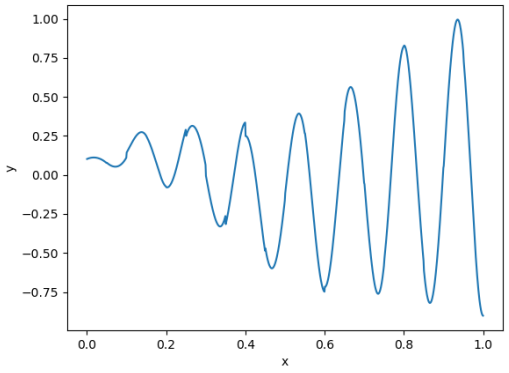


(b) Split DG Net

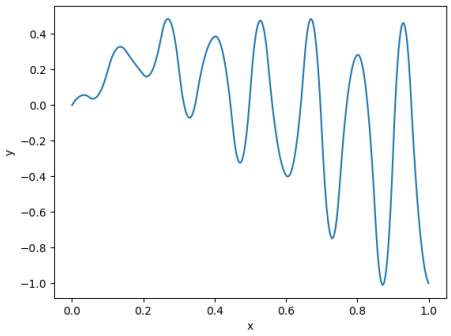
当 $w = 15\pi$ 时, 高频解：



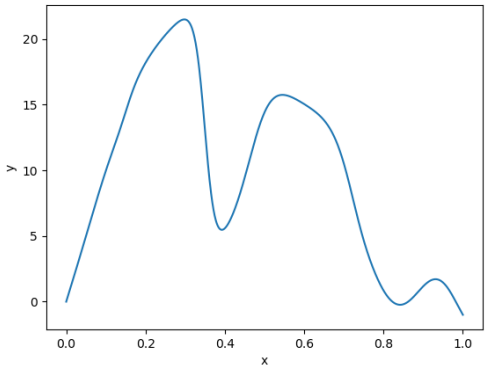
(a) Exact



(b) Split DG Net

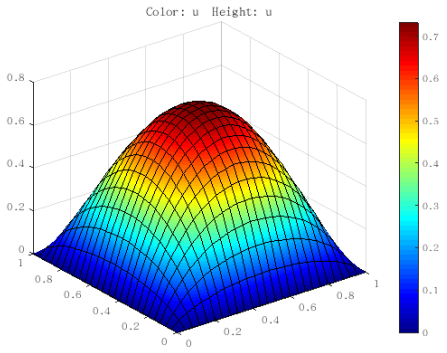


(c) Split PWNN

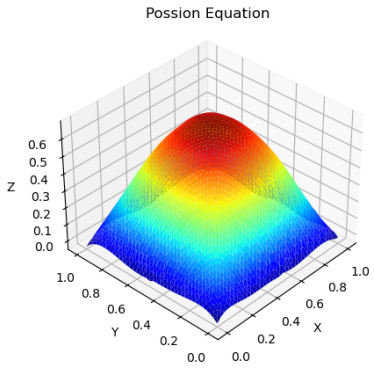


(d) PWNN

2 2D Possion



(a) Split DG Net



(b) Split DG Net

上处图像并非由 PWNN 官方程序生成，因此无论是训练速度和训练精度可能和官方有些差距。代码还有很大的优化空间。不幸的是因为时间原因，只能到此地步。

六、总结与展望

1 工作总结

在本项工作中，我们主要探讨了经典的间断伽辽金方法，主要是以内罚形式实现。在此基础上，我们将间断有限元和时空方法结合，基于热方程以及 NS 方程介绍其理论，并完成了相关的数值算例，证明了经典算法的可行性。在此基础上我们从另外一个角度看待了有限元类方法，将深度学习的方法和 DG 方法相结合，在现有处理弱形式方程的基础上进行改进，提出了新的思路，并通过了相关的数值算例，证明了其方法的可靠性。

DGNet 其实本质上是一种自适应的想法。在一些没有出现复杂解的情况下其实没必要将网络拆开。这也就是为什么间断有限元在求解有间断解等问题上展现出优势，而在一般的问题上用间断方法其实增加了计算量。也就是说当网络的非线性性足够去拟合一个方程的解的时候，没有必要将网络拆分成小的网络。因为可以看到为了小网络之间的耦合，我们添加了两个惩罚项在损失函数中作为约束，这其实增加了优化的难度。因此当网络很大，比如单个拥有 4 层 50 个神经元的隐藏层网络不足以拟合某个，例如有高频振荡，或者间断解的问题时，可以采用 DGNet 的方法。

2 研究展望

首先值的说明的第一是实验代码还有很大的优化空间，现在的训练速度还不够快，只是对于一些基本问题方案体现了其合理性。第二 DG 方法主要的优势在与间断跳跃解，在局部存在奇点的情况。这些情况在本项工作中还没有体现，而仅仅完成了初步的实验。因此对于实验还有很多需要完善的地方，无论是和经典的 PINN 等方法的精度和训练速度对比。第三关于将 Split DG Net 应用到时空有限元方法上应该是很自然的，但是这部分实验还未来得及实践。

但我仍然认为这是一个非常值的继续研究的方法，尤其是在完成在时空有限元方法上的实践之后，很自然的可以应用到 1. 动边界的处理上面，在工业上例如半导体制造中的薄膜沉积的仿真上；2. 区域分解，在不同区域进行不同的离散化之后可以实现微观和宏观部

分的耦合，是多尺度所关心的问题。

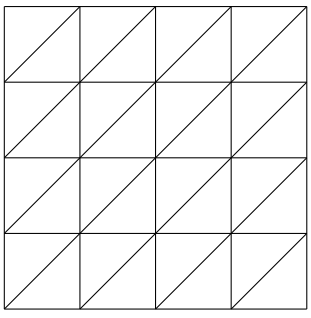
还有一些思路还未来得及尝试，在未来工作中可以再去具体实现：

数据监督

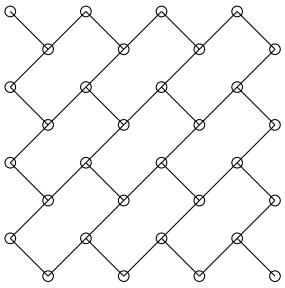
其实，深度学习很大一部分优势还在于数据。因此当我们拥有数据之后，可以让优化变得更加简单。在上述的工作中，最终的约束损失函数中有四项，这对于没有数据只进行优化的深度学习任务来说是很困难的。但是当我们拥有数据后，我们不仅在局部可以拆分成拥有边界信息的小块，从而进行训练，而且机理也可以部分未知，实现反问题的求解。当局部数据给定之后，由于单元上网络的独立，可以通过固定除了单元以及单元邻居外的网络的参数，仅仅进行局部的参数调整来加快训练速度。

图神经网络

图神经网络近些年也展现出它强大的威力，尤其是在处理图形数据的时候。然而对于三角化之后的离散区域，我们很容易发现，三角单元之间的关系就是图状结构，并且有封闭的性质，如图 2 维可以抽象为：

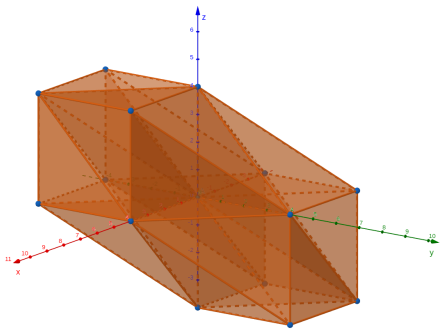


(a) Triangluation

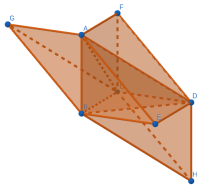


(b) Graph of cells

三维可以表示为：



(a) Neighbour nodes of center



(b) Cells to Nodes

并且我们可以发现，在 DG 方法中定义的跳量以及平均都可以储存在边上。我们也可以发现，在 DG 方法中，我们装配的刚度矩阵和图神经网络中的邻接矩阵有很多的共同点，并且已经有工作涉及 DG 方法和 GNN 融合，但还是聚焦在如何求解刚度矩阵的线性方程组。

3 参考文献

[1] RIVIÈRE B. Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations[M/OL]. Society for Industrial, 2008. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898717440. https://epubs.siam.org/doi/abs/10.1137/1.9780898717440. DOI: 10.1137/1.9780898717440.

[2] 武海军. 偏微分方程数值方法[M]. 2023.

[3] NEUMÜLLER M. Space-Time Methods:Fast Solvers and Applications[M]. 2013.

[4] RAISSI M. Deep hidden physics models: deep learning of nonlinear partial differential equations[J]. J. Mach. Learn. Res., 2018, 19(1): 932-955.

[5] GAO H, SUN L, WANG J X. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving Parameterized Steady-State PDEs on irregular domain[J/OL]. Journal of Computational Physics, 2020: 110079 [2021-01-04]. https://linkinghub.elsevier.com/retrieve/pii/S0021999120308536. DOI: 10.1016/j.jcp.2020.110079.

[6] WANDEL N, WEINMANN M, KLEIN R. Learning Incompressible Fluid Dynamics from Scratch - Towards Fast, Differentiable Fluid Models that Generalize[C]//Ninth International Conference on Learning Representations. 2021.

[7] LU L, JIN P, PANG G, et al. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators[J]. Nature Machine Intelligence, 2021, 3(3): 218-229.

[8] LI Z, KOVACHKI N, AZIZZADENESHELI K, et al. Fourier Neural Operator for Parametric Partial Differential Equations[J]., 2020.

[9] EE W, YU B. The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems[J]. Communications in Mathematics and Statistics, 2017, 6. DOI: 10.1007/s40304-018-0127-z.

[10] ZANG Y, BAO G, YE X, et al. Weak adversarial networks for high-dimensional partial differential equations[J/OL]. Journal of Computational Physics, 2020, 411: 109409. https://www.sciencedirect.com/science/article/pii/S0021999120301832. DOI: https://doi.org/10.1016/j.jcp.2020.109409.

[11] YAO H, GAO Y, LIU Y. FEA-Net: A physics-guided data-driven model for efficient mechanical response prediction[J/OL]. Computer Methods in Applied Mechanics and Engineering, 2020, 363: 112892. https://www.sciencedirect.com/science/article/pii/S0045782520300748. DOI: https://doi.org/10.1016/j.cma.2020.112892.

[12] ZANG Y, BAO G. ParticleWNN: a Novel Neural Networks Framework for Solving Partial Differential Equations[Z]. 2023. arXiv: 2305.12433 [cs.LG].

附录

DG 方法解的存在性和唯一性

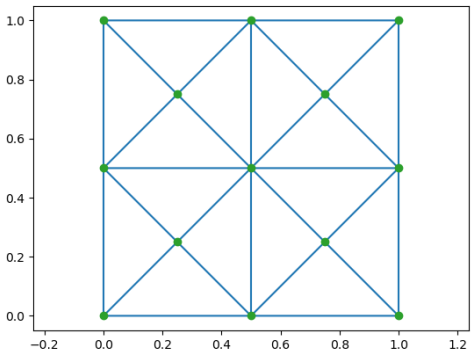
假设 (i), (ii), (iii) 中任意一条成立:

- (i) 若形式为 NIPG, $k \geq 1$ 且 $\alpha > 0$ 或 $\sigma^0 > 0$;
- (ii) 若形式为 SIPG 或 IIPG, $k \geq 1$ 且 σ^0 有界，被一个常数控制;
- (iii) 若形式为 NIPG, $k \geq 2$ 且 $\sigma^0 = 0$, $\alpha = 0$.

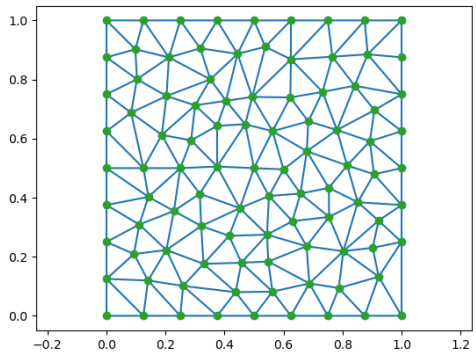
那么 DG 方法的解 u_h 存在且唯一。

Triangle

该 python 库可以实现 2D 三角化，并且能够通过设置边界连接方式，单元最大面积以及最大角度来改变三角化参数，如下图。

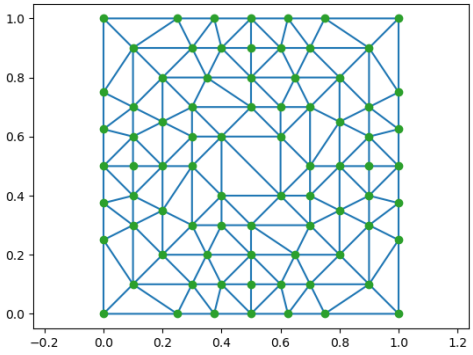


(a) 角度不小于 30, 面积 0.1

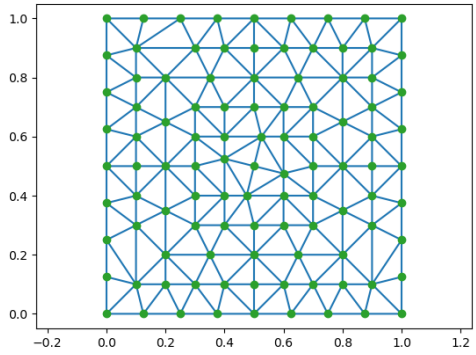


(b) 角度不小于 30, 面积 0.01

图 4.1: SIPG



(a) 内部设置强制边界，角度不小于 30, 面积 0.1



(b) 内部设置强制边界，角度不小于 30, 面积 0.01

图 4.2: Triangle

代码

相关代码上传到https://gitee.com/Zebrainy-cgy/dgnet.git

作者简历

姓名陈冠宇，性别男，民族汉，籍贯江苏淮安。于 2020 年考入浙江大学，在数学科学学院数学与应用数学 (强基计划) 就读，拟于 2024 年六月本科毕业。