G·格子达论文检测报告【全文标注】

报告编号:262D4F9820534B66BB4FBAC228776C1C

送检文档:带间断系数的弹性问题的有限元方法

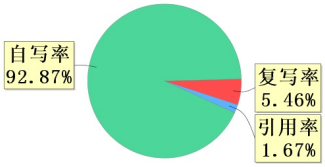
作 者:唐小康　　送检单位:湘潭大学　　送检时间:2023-05-23 17:25:45

比对索引库

1989-01-01至2023-05-23

学术期刊库	报纸资源库	本科论文共享库	格子达公识库
学位论文库	互联网资源库	专利库	机构自建库
会议论文库	格子达多元库		

检测结果



总相似比:7.13%(总相似比=复写率+引用率)

查重检测指标:自写率92.87%复写率5.46%引用率1.67%（含自引率0.0%）

其他类型检测结果:去除引用后总相似比:5.46% 同校同届总相似比:0.96%

相似片段:复写片段32 同届片段7 引用片段8

指标名称	学校要求	指标检测结果	系统判定
总相似比	不超过20%	7.13%	符合
复写率	不超过20%	5.46%	符合
同届比	不超过20%	0.96%	符合
论文总字数	不少于800字/单词	35155字符	符合

其他检测结果：

指标名称	识别数量	系统判定
代码块检测	1	--

疑似书写不规范引用汇总

引用片段	错误描述
当研究的弹性体形状和受力具有一定特点时，通过适当的简化处理，就可以归结为平面弹性问题[1]	未发现一对一的引用注释
对于各向同性均匀介质的平面弹性问题，当材料的 Lame 常数 λ 时，即对于几乎不可压介质，通常低阶的协调有限元解，往往不再收敛到原问题的解，或者达不到最优收敛阶，这就是闭锁现象[2]	未发现一对一的引用注释
根据型函数建立过程中是否需要网格剖分，这些数值方法可以分为两类：一类是有网格方法，这其中包括高阶有限元法[3]、混合有限元法[4]、增强有限元法[5]和不连续 Galerkin 法[6]等	未发现一对一的引用注释
另一类是无网格方法，无网格方法又可分为弱形式无网格法和强形式的无网格方法[1]	未发现一对一的引用注释
具有紧致支集的函数必在边界 Γ 的某一邻域内恒等于零[7]	未发现一对一的引用注释
过 p 点作与三个顶点的连线，将(p0, p1, p2) 分成三个三角形: (p1, p2, p), (p0, p, p2), (p0, p1, p), 其面积分别为 S0, S1, S2.[7]	未发现一对一的引用注释
这就是所谓的锁定现象[9]	未发现一对一的引用注释
则存在一个与 h 无关的正常数 C 使得[9]	未发现一对一的引用注释

复写相似文献列表

序号	相似文献	相似片段数	相似字数	相似比
1	篇名:《本科联盟c9e67c987f2b52abfe00a8852c8cf2f2》 来源:大学生本科毕业论文联合共享库 出处: 2022	8个	268	1.37%
2	篇名:《本科联盟496f4755fd249a3574b40ae12e07a989》 来源:大学生本科毕业论文联合共享库 出处: 2022	5个	224	1.15%
3	篇名:《（近）不可压缩平面弹性问题的位移-压力混合重心插值配点法》 来源:学位论文库 作者:徐子康 出处:硕博学位论文 2018	2个	70	0.36%
4	篇名:《某雷达接收系统结构设计与分析研究》 来源:学位论文库 作者:刘国维 出处:硕博学位论文 2013	1个	64	0.33%
5	篇名:《电动工具振动与噪声研究》 来源:学位论文库 作者:孙辉 出处:硕博学位论文 2012	1个	51	0.26%
6	篇名:《本科联盟f1f6431aa49e0be9eaab4a24bb9f0989》 来源:大学生本科毕业论文联合共享库 出处: 2022	1个	35	0.18%
7	篇名:《某型号仪器盒结构强度与振动仿真研究》 来源:学位论文库 作者:邹金红 出处:硕博学位论文 2018	1个	35	0.18%
8	篇名:《路堤荷载下带帽桩—网复合地基力学性状有限元分析》 来源:学术期刊库 作者:高胜利 魏宏 刘天福 出处:高胜利 魏宏 刘天福 2010	1个	31	0.16%
9	篇名:《对流扩散方程的特征有限元方法》 来源:学位论文库 作者:王晓玲 出处:硕博学位论文 2007	1个	31	0.16%
10	篇名:《基于ERP和CRM整合的销售执行系统分析与设计》 来源:学位论文库 作者:尹纯峰 出处:硕博学位论文 2013	1个	32	0.16%

序号	相似文献	相似片段数	相似字数	相似比
11	篇名:《本科联盟2049a5841ffb0003077db6e926d2b644》 来源:大学生本科毕业论文联合共享库 出处: 2018	1个	29	0.15%
12	篇名:《基于NURBS的样条有限元法研究》 来源:学位论文库 作者:岳东风 出处:硕博学位论文 2014	1个	29	0.15%
13	篇名:《本科联盟6dc6f846c8a04e1c07b38d3363cff62f》 来源:大学生本科毕业论文联合共享库 出处: 2022	1个	27	0.14%
14	篇名:《优化软件系统SIPOPT设计与实现及其在工程中的应用》 来源:学位论文库 作者:杨春峰 出处:硕博学位论文	1个	25	0.13%
15	篇名:《本科联盟56471387fe4344f6db06680ef9dc5070》 来源:大学生本科毕业论文联合共享库 出处: 2021	1个	25	0.13%
16	篇名:《基于可靠度的土石坝风险分析及维修策略研究》 来源:学位论文库 作者:刘兴芳 出处:硕博学位论文 2009	1个	22	0.11%
17	篇名:《三轴转台结构静动态特性分析与动力学仿真 》 来源:学位论文库 作者:潘健 出处:硕博学位论文 2009	1个	22	0.11%
18	篇名:《松散破碎地层泵吸式孔底局部反循环取心钻具研究》 来源:学位论文库 作者:陈宗涛 出处:硕博学位论文	1个	20	0.1%
19	篇名:《两个轮子一起转——浅谈如何开拓农村商品市场》 来源:学术期刊库 作者:饶生来 郑胜利 出处:饶生来 郑胜利 1999	1个	16	0.08%
20	篇名:《本科联盟ea6bcabe9588f9b55b57b48b9ba38fed》 来源:大学生本科毕业论文联合共享库 出处: 2020	1个	9	0.05%

同届相似文献列表

仅做参考，同届相似比值不计入总相似比

序号	学校	院系专业	姓名	账号	相似文献	相似片段数	相似字数	相似比
1	湘潭大学	数学与计算科学学院信息与计算科学	盛政宇	201905755616	篇名:二维椭圆方程反问题的有限元方法	3个	76	0.39%
2	湘潭大学	机械工程学院工程力学	黄肖依	201905800802	篇名:纤维增强高分子复合材料的力学性能分析	1个	55	0.28%
3	湘潭大学	数学与计算科学学院信息与计算科学	柯宇鹏	201905755521	篇名:基于有限元离散的椭圆界面问题的快速迭代法	1个	27	0.14%
4	湘潭大学	数学与计算科学学院数学与应用数学	肖凯	201905556021	篇名:一个新的Bernstein不等式和二维耗散拟地转方程	1个	15	0.08%

序号	学校	院系专业	姓名	账号	相似文献	相似片段数	相似字数	相似比
5	湘潭大学	数学与计算科学学院信息与计算科学	王淦	201805750404	篇名:线性代数方程组求解的几类迭代方法及其应用	1个	15	0.08%

全文内容

湘潭大学毕业论文

题目：带间断系数的弹性问题的有限元法

学院： 数学与计算科学学院

班	级：2019 信息与计算科学 2 班	
学	号：	201905755601
姓	名：	唐小康
指导教师：		王华
完成日期：		2023 年 5 月

湘潭大学

毕业设计说明书

题目：带间断系数的弹性问题的有限元法

学院： 数学与计算科学学院

专	业：	信息与计算科学
学	号：	201905755601
姓	名：	唐小康
指导教师：		王华
完成日期：		2023 年 5 月

湘潭大 学

毕业论文（设计）任务书

论文（设计）题目：		带间断系数的弹性问题的有限元法			
学号：	201905755601	姓名：	唐小康	专业：	信息与计算科学
指导教师：		系主任：			

一、主要内容及基本要求

主要内容：

1、使用非协调有限元求解平面弹性问题；

2、使用非协调有限元求解带间断系数的平面弹性问题。

基本要求：

1、选题体现专业特点及教学要求，难度和份量适中；

2、搜集、精读、归纳与选题有关的文献不少于 10 篇；

3、撰写研究综述的现状时要有学术观点提炼和文献引用，并进行辩证评析；4、要求论文结构合理，内容充实，论据可靠，论证有力，主题明确，概念准确，层次清楚，重点突出，文字简练，文理通顺，有自己的见解；

5、论文中的各级大小标题文字一律要经过提炼；

6、必要时论文中要有注释，并采用尾注的形式；

7、按照毕业论文的各项进度要求完成工作，主动并及时与指导老师联系；8、毕业论文的正文篇幅不少于 8,000 字。

二、重点研究的问题

1、有限元的近似

2、弹性问题闭锁现象及其解除

三、进度安排

序号	各阶段完成的内容	完成时间
1	毕业论文选题	2022 年 9 月
2	围绕选题搜集、阅读、整理文献资料	2022 年 9 月
3	开题：完成文献综述和论文框架	2022 年 11 月
4	中期检查：提交论文初稿	2023 年 3 月中旬
5	论文查重检测	2023 年 4 月
6	论文定稿后排版打印、交稿	2023 年 5 月
7	毕业论文答辩	2023 年 5 月
8	答辩后的论文修改	2023 年 5 月下旬

四、应收集的资料及主要参考文献

1、李荣华. 偏微分方程数值解, 2007.

2、陈纪修, 於崇华, 金路. 数学分析. 高等教育出版社, 2004.

3、邵文婷. 求解一类交界面问题的模态基函数谱元法数值实验. 上海第二工业大学学报, 34(4):283290, 2017.

4、王兆清, 徐子康, 李金. 不可压缩平面问题的位移-压力混合重心插值配点法. 应用力学学报, 35(3):631636, 2018.

5、Susanne C Brenner, L Ridgway Scott, and L Ridgway Scott. The mathe-matical theory of finite element methods, volume 3. Springer, 2008.

湘潭大学

毕业论文（设计）评阅表

学号： 201905755601姓名：唐小康专业：信息与计算科学

毕业论文（设计）题目：带间断系数的弹性问题的有限元法

评价项目	
选题	1.是否符合培养目标，体现学科、专业特点和教学计划的基本要求，达到 综合训练的目的； 2.难度、份量是否适当； 3.是否与生产、科研、社会等实际相结合。
能力	1.是否有查阅文献、综合归纳资料的能力； 2.是否有综合运用知识的能力； 3.是否具备研究方案的设计能力、研究方法和手段的运用能力；4.是否具备一定的外文与计算机应用能力； 5.工科是否有经济分析能力。
论文 （设计）质量	1.立论是否正确，论述是否充分，结构是否严谨合理；实验是否正确，设 计、计算、分析处理是否科学；技术用语是否准确，符号是否统一，图表 图纸是否完备、整洁、正确，引文是否规范； 2.文字是否通顺，有无观点提炼，综合概括能力如何； 3.有无理论价值或实际应用价值，有无创新之处。

综合评价该生论文选题符合专业培养目标，能够达到综合训练目标，文章篇幅 符合学院规定，内容较为完整，实证分析较为科学，态度较为认真，参考 了丰富的文献资料，有一定的个人见解

同意其参加答辩

评阅人：

2023 年 5 月 15 日

湘潭大学

毕业论文（设计）鉴定意见

学号：201905755601姓名：唐小康专业：信息与计算科学

论文（设计）题目：带间断系数的弹性问题的有限元法

内容提要

本文以带间断系数的弹性问题为研究对象首先介绍了研究背景和国内外研究现 状，然后回顾了有限元理论的基

本知识，包括 Sobolev 空间、弹性问题的边值问题和 变分、带间断系数的方程的引入、离散方法和误差估计等 接着给出了几个算例，分 别展示了弹性问题和带间断系数的弹性问题在有限元方法下的数值解，比较了不同的 参数对结果的影响最后总结了本文的主要结论，指出了存在的不足和未来的研究方 向

指导教师评语 论文选题符合专业培养目标，全文结构基本合理科学，逻辑思路清晰，观点表达 准确，语言流畅，论证方法较合理，参考的文献符合主题要求，从主题到内容符合专 业要求，衔接的比较紧密，但个别引文没有标著出来，真正属于自己创新的内容还不 是很多，个别概念比较模糊，总体上达到毕业论文要求。 指导教师： 2023 年 5 月 10 日
答辩简要情况及评语 该生能较好地运用所学理论和专业知识，按期圆满地完成论文，有一定地独立 工作能力，但在完成任务书所要求地设计内容上有欠缺。论文分析合理，层次分明，文字通畅，论文格式基本符合要求。 答辩小组组长： 2023 年 5 月 20 日
答辩委员会意见 答辩委员会主任： 2023 年 5 月 22 日

摘要

本文以带间断系数的弹性问题为研究对象。首先介绍了研究背景和国内外研究现状，然后回顾了有限元理论的基本知识，包括 Sobolev 空间、弹性问题的边值问题和变分、带间断系数的方程的引入、离散方法和误差估计等。接着给出了几个算例，分别展示了弹性问题和带间断系数的弹性问题在有限元方法下的数值解，比较了不同的参数对结果的影响。最后总结了本文的主要结论，指出了存在的不足和未来的研究方向。

关键字：平面弹性问题; 间断系数；非协调有限元; locking-free

Abstract

This paper studies the elastic problem with discontinuous coefficients. Firstly, the research background and the domestic and foreign research status are intro-duced. Then, the basic knowledge of finite element theory is reviewed, including Sobolev space, boundary value problem and variational form of elastic problem, introduction of equation with discontinuous coefficients, discrete method and error estimation. Next, several examples are given to show the numerical so-lutions of elastic problem and elasticity problem with discontinuous coefficients under finite element method, and the influence of different parameters on the results is compared. Finally, the main conclusions of this paper are summarized, and the existing shortcomings and future research directions are pointed out.

Key words： linear elasticity problem; discontinuous coefficient；noncon-forming finite element; locking-free

目录

1 绪论

1.1 研究背景

带间断系数的方程是指一种用于模拟复合材料层合板的力学性能的数学模型。复合材料层合板是由两层或多层单层板粘合在一起的结构单元，具有不同的铺层顺序和铺层角度，可以表现出不同的力学特性。带间断系数的方程是一种考虑了层间应力和层间变形协调的方程，可以描述层合板在面内和面外的应力-位移关系。带间断系数的方程也可以用于模拟材料的分离和剥落，例如复合材料剥离、金属焊接材料损伤、混凝土材料开裂等。

平面弹性力学方程组是弹性力学中最基础、最常见的模型。当研究的弹性体形状和受力具有一定特点时，通过适当的简化处理，就可以归结为平面弹性问题[1]。对于各向同性均匀介质的平面弹性问题，当材料的 Lame 常数 λ 时，即对于几乎不可压介质，通常低阶的协调有限元解，往往不再收敛到原问题的解，或者达不到最优收敛阶，这就是闭锁现象[2]。

为了消除（近）不可压缩弹性问题中遇到的闭锁现象，国内外研究学者提出了多种有效的数值分析方法。根据型函数建立过程中是否需要网格剖分，这些数值方法可以分为两类：一类是有网格方法，这其中包括高阶有限元法[3]、混合有限元法[4]、增强有限元法[5]和不连续 Galerkin 法[6]等；另一类是无网格方法，无网格方法又可分为弱形式无网格法和强形式的无网格方法[1]。

1.2 国内外研究现状

有限元法是一种的数值分析方法，它可以用来求解各种复杂的工程问题。有限元法的核心思想最早可以追溯到 1943 年，当时 R.W.Courant 提出了一种基于变分原理的离散化方法。1956 年，R.W.Clough 等四位教授与工程师在一篇发表在科技期刊上的论文中，首次将这种方法应用到飞机机翼强度的计算中，并将其命名为刚性法(Stiffness)。这篇论文标志着有限元法在工程学界上的正式诞生。在 1960 年，R.W.Clough 教授在一篇关于平面弹性问题的论文中，首次提出了“有限元法”这个术语，并将这种方法应用到了土木工程领域。三年后，Richard MacNeal 博士与 Robert Schwendler 合作创立了 MSC 公司，并开发出了一款名为 SADSAM 的软件程序，实现了数字仿真模拟结构分析的功能，这标志着有限元方法 (FEA) 从理论走向了实践。1964-1965 年期间，O.C.Zienkiewicz 等人在多篇论文中，采用极小位能原理和虚功原理，以一种新颖的思路推导出了有限元法。

在我国，有限元方法的发展历史上，涌现出了一批杰出的学者，他们为有限元方法的理论和应用做出了重要的贡献，如冯康（有限单元法理论），陈伯屏（结构矩阵方法），钱令希（余能定理），钱伟长（广义变分原理）等。但是，受到当时的国际和国内环境的制约，我国的学者在有限元方法的深层次研究上遇到

了很多困难，很难跟上国际的发展步伐，导致与国外的技术水平之间的差距逐步扩大。20 世纪 60 年代初期，我国的老一辈计算科学家较早地将计算机应用

于土木、建筑和机械工程领域。当时黄玉珊教授就提出了“小展弦比机翼薄壁结构的直接设计法”和“力法—应力设计法”；而在 70 年代初期，钱令希教授提

出了“结构力学中的最优化设计理论与方法的近代发展”。这些理论和方法都为国内的有限元技术指明了方向。1964 年初崔俊芝院士研制出国内第一个平面问题通用有限元程序，解决了刘家峡大坝的复杂应力分析问题。20 世纪 60 年代到 70 年代，国内的有限元方法及有限元软件诞生之后，曾计算过数十个大型工程，应用于

水利、电力、机械、航空、建筑等多个领域。20 世纪 70 年代中期，大连理工大学研制出了 JEFIX 有限元软件，航空工业部研制了 HAJIF 系列程序。80 年代中期，北京大学的袁明武教授通过对国外 SAP 软件的移植和重大改造，研制出了 SAP-84；北京农业大学的李明瑞教授研发了 FEM 软件；建筑科学研究院在国家“六五”攻关项目支持下，研制完成了“BDP-建筑工程设计软件包”；中国科学院开发了 FEPS、SEFEM；航空工业总公司飞机结构多约束优化设计系统 YIDOYU 等一批自主程序。

然而，在上世纪 90 年代，国外的有限元软件大规模地进入国内市场，涵盖了各个领域。国外的学者专家也经常到各大学、工厂和企业进行技术推广和使用指导，使得国内有限元方法的发展面临着更大的挑战。管理部门对有限元软件的认识也出现了偏差，对此缺少必要的支持，核心技术控制在国外，所以一直到上世纪末期，国内自主技术创新的速度十分缓慢。但是，在 21 世纪初期以来，国内拥有自主知识产权的软件逐渐实现了市场化，取得了一定的发展空间，同时也引起了国家对有限元技术的高度重视，使得有限元方法逐渐走出低迷状态，不再仅仅停留在高校和企业之中。

2

2 有限元理论

2.1 Sobolev 空间

假定 G 是有界平面区域，其边界 Γ 是按段光滑的简单闭曲线，G = G Γ 是G 的闭包。对于 G 上的任一函数 u(x, y), 称集合 {(x, y) | u(x, y) = 0, (x, y) G} 的闭包为 u 的支集。如果 u 的支集 G 内，则说 u 于 G 具有紧致支集。具有紧致支集的函数必在边界 Γ 的某一邻域内恒等于零[7]。

用 C0表示 G 上无穷次可微并具有紧致支集的函数类，L2(G) 是定义在 G 上平方可积的可测函数空间，其内积和范数分别为

(f, g) =Gfgdxdy,||f|| =(f, f) = (G |f|2dxdy)对 f L2(G), 如果存在 g, h L2(G), 使等式1

2 .(2.1.1)

(2.1.2)

对任意的 φ C数 h, 记作

定义

∫ G gφdxdy = ∫ g f φx dxdy, (2.1.3) ∫ G hφdxdy = ∫ G f φy dxdy, (2.1.4) 成立，则说 f 对 x 的一阶广义导数 g 和对 y 的一阶导	
fx =fx= g,	(2.1.5)
fy =fy= y.	(2.1.6)

H1(G) = {f(x, y)|f, fx, fy L2(G)}, 其中 fx, fy 是 f 的广义导数。与 H1(G) 引入内积

(f, g)1 =G[fg + fxgx + fygy]dxdy,1

2 .(2.1.7)

和范数则 H1() 是 Hilbert 空间，称之为 Sobolev 空间||f||1 =(f, f) = (G [|f|2+ |fx|2+ |fy|2]dxdy)(2.1.8)

3

2.2 弹性问题

2.2.1 边值问题

令 u，g，t，σ = (σij)1i,j2，τ = (τij)1i,j2 是双变量函数，定义以下符号

(u) =1 2(grad u + (grad u)t),

tr(τ) = τ11 + τ22,grad(u) = (u2xxu2 yy u1u1 δ = (10 01) ,	)	,

div u =u1x+ u2y,div τ = (τ 11 τ 12 x+ τ 12x+ τ 22)	,
i=1 2 ∑ j=1 2 ∑ σ : τ = σ ij τ ij. 考虑各项同性弹性材料，令 u(x, y), f(x, y) 是其位移和体力，由线弹性问题	

的静态理论，u, f 满足以下方程

div σ (u) = f ,(2.2.1)

应力张量 σ (u) 定义为

σ (u) = 2(u) + λ tr((u)) δ . (2.2.2)

其中 R2, 正常数 λ , 为 Lame 常数。假定 (, λ) [1, 2] × (0, +).

Γ 1 上的位移边界条件为令 Γ 1、Γ 2 为 的两个开子集，使得 = Γ 1 Γ 2 并且 Γ 1 Γ 2 = , 令

u| Γ 1 = g. (2.2.3)

并且 Γ 2 上的牵引力边值条件为

(σ (u) ν)| Γ 2 = t. (2.2.4)

如果 Γ 1 = (或 Γ 2 =), 则边值问题为纯牵引力 (或纯位移) 问题。

2.2.2 变分

对于齐次纯位移问题，令 u 在边界上满足

u| = 0. (2.2.5)

设 ν = (ν 1, ν 2)t, ν 1, ν 2 C0(), 方程 (2.2.1) 两边同乘 ν 并积分得

4

参考文献知[8](div σ (u)) ν dxdy =f ν dxdy.(2.2.6)

(2.2.7)

(2.2.8)

fdiv a = div (fa) a : grad f, div a dV = a dS.将边界条件 (2.2.5)，方程 (2.2.7)，(2.2.8) 带入方程 (2.2

.6) 得(div σ (u)) ν dxdy= div (σ (u) ν) dxdy σ (u) : grad ν dxdy= Γ σ (u) ν dxdy + σ (u) : grad ν dxdy=

σ (u) : grad ν dxdy= 2(u) : grad ν + λ div u div ν dxdy= grad u : grad ν dxdy + (+ λ) div u div ν dxdy.

所以

grad u : grad ν dxdy + (+ λ)div u div ν dxdy =f ν dxdy.(2.2.9)

该问题的变分问题为，求 u H1() 使得 u| Γ 1 = 0, 并且 a(u, ν) = f · ν dxdy ν V,(2.2.10)

其中

a(u, ν) : = grad u : grad ν dxdy + (+ λ)div u div ν dxdy,(2.2.11)

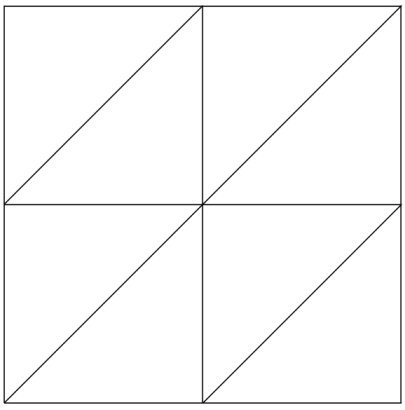
V : = { ν H1()| ν | Γ = 0}.

Lax-Milgram 定理[9]：设 H 是 Hilbert 空间，a(· , ·) 是 H × H 上的有界的强制的双线性泛函。则对任意的 F H, 存在唯一的 u H 满足

a(u, ν) = (f, ν), ν H.(2.2.12)

由 Lax-Milgram 定理知，此变分问题的解存在且唯一。

5



2.2.3 引入间断系数

设 1，2 是 的两个子集，使得 1

考虑以下边值问题 2 = 并且 1 2 = ,

div σ (u) = f , (2.2.13)

u| = 0.

当 Lame 常数 λ , 在 1, 2 上取不同值，即 (x, y) 1 时 λ = λ 1, =

1, (x, y) 2 时 λ = λ 2, = 2, 并且 λ 1 = λ 2, 1 = 2, 通过计算得到与此问题对应的双线性形式为

a(u, ν) = 11 grad u : grad ν dxdy + (1 + λ 1)1 div u div ν dxdy (2.2.14)

+ 22 grad u : grad ν dxdy + (2 + λ 2)2 div u div ν dxdy.

2.3 离散

2.3.1 Galerkin 法

p3 p8 p2

1 5

7 2

p6

p034p560p7

p4p1

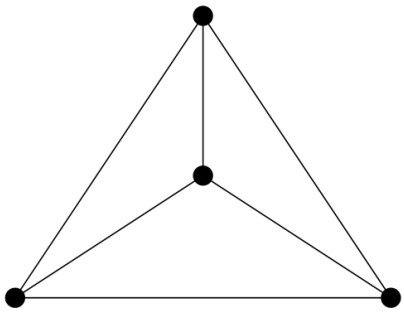
图 1

设求解区间 = [0, 1] × [0, 1], 首先对其按照图 1 进行网格剖分， 节点为 p0, p1, ..., pn.

图中的三角形区域称为单元。

其次，在 Sobolev 空间 H1内取子空间 Uh, 它的元素在每一单元是次数不超过某一正整数 m 的多项式，在全区域 上属于函数空间 H1. 则 Uh × Uh 为试探函数空间。

6



设

Uh = span(φ0, φ1, ..., φn),

2i = (φi, 0),2i+1 = (0, φi),i = 0, ..., n.

则 uh Uh × Uh, 可表成

uh =2n+1cii.(2.3.1)

将式 (2.3.1) 带入方程 (2.2.10) 中得到 Galerkin 方程

2n+1

a(i, j)ci = (f, j),j = 0, 1, ..., 2n + 1.(2.3.2)

令

A = (a(j, i))0i,j2n+1,

F = ((f, i))0i2n+1,

c = (ci)0i2n+1.

则 Galerkin 方程 (2.3.2) 的矩阵形式为

Ac = F. (2.3.3)

考虑齐次边界条件，若 (xi, yi) 为边界点，则 A 第 2i 行第 2i 列，第 2i + 1 行第 2i + 1 列元素为 1, 第 2i 和 2i + 1 行的其他元素及 F(2i), F(2i + 1) 都为 0.

2.3.2 线性元

如图 2，设 (p0, p1, p2) 是以 p0, p1, p2 为顶点的任意三角型元，面积为 S. 在 (p0, p1, p2) 内任取一点 p, 坐标为 (x, y). 过 p 点作与三个顶点的连线，将(p0, p1, p2) 分成三个三角形: (p1, p2, p), (p0, p, p2), (p0, p1, p), 其面积分别为 S0, S1, S2.[7]

p2

p

p0p1

图 2

7

显然 S0 + S1 + S2 = S, 令

L0 =S0 S ,			L1 =S1 S , L2 =S2 S ,	(2.3.4)
因为	L0 =L1 =L2 =	1 2S[(x2y3 x3y2) + (y2 y3)x + (x3 x2)y],1		(2.3.5)
L0 =1, 0, 0,L1 =0, 1, 0,L2 =0, 0, 1, 所以在此区间上 φi = Li, 即				
x = x0, y = y0, x = x1, y = y1, x = x2, y = y2, x = x0, y = y0, x = x1, y = y1, x = x2, y = y2, x = x0, y = y0, x = x1, y = y1, x = x2, y = y2,	2.3.3	φ0 =φ1 =φ2 =C-R 元	1 2S[(x2y3 x3y2) + (y2 y3)x + (x3 x2)y],1 2S[(x3y0 x0y3) + (y3 y0)x + (x0 x3)y],1 2S[(x0y1 x1y0) + (y0 y1)x + (x1 x0)y].	

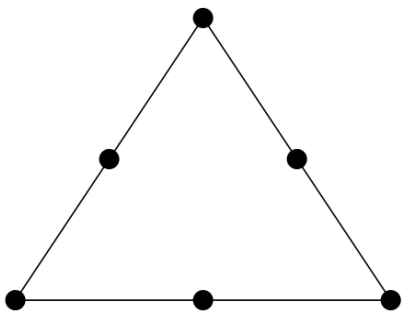
如图 3，设三角形 (q0, q1, q2) 是以 q0, q1, q2 为顶点的任意三角形元，p0, p1, p2 为其三条边的中点，其坐标分别为 (x0.y0), (x1, y1), (x2, y2).

设三角形 (q0, q1, q2) 上的 C-R 元为 φ0, φ1, φ2,

φi = aix + biy + ci,i = 0, 1, 2,(2.3.6)

且其在 p0，p1，p2 点上满足以下关系式

φi(pj) =	1,0,	i = j i = j 8	,	i, j = 0, 1, 2.	(2.3.7)



q0

q1p2p1q2

p0

图 3

设

A =x0x1 y0y1 1 1ci = (ai, bi, ci)t,

则方程组 (2.3.7) 的矩阵形式为x2 y2 1 ,

f = (x, y, 1)t.

Aci = ei,i = 0, 1, 2.(2.3.8)

通过计算可以得到单元 (q0, q1, q2) 上的 C-R 元为

2.4误差估计 φi = A1eif,i = 0, 1, 2.(2.3.9)

假设 是一个凸多边形区域，并且 Γ 1 or Γ 2 中任意一个为空。对于纯位移问题 (Γ 2 =), 只考虑齐次边界条件。

令 Th是 三角划分的一个非退化族。对于纯位移问题 (Γ 2 =), 使用有限元空间

Vh := { ν H1() : ν |T , T Th}, 并且对于纯牵引力问题 (Γ 1 =), 使用

Vh := { ν H1() : ν |T , T Th},

令 u H2() H1() 满足纯位移问题，并且 uh Vh 满足 a(uh, ν) = f · ν dx ν Vh.(2.4.1)

则存在一个正常数 C(, λ) 使得[9]

u uhH1() C(, λ)huH2(). (2.4.2)

令 u H2() 满足纯牵引力问题令 uh Vh 满足a(uh, ν) = f · ν dx + Γ 2 t · ν ds ν Vh.(2.4.3)

则存在一个正常数 C(, λ) 使得[9]

2.5闭锁现象u uhH1() C(, λ)huH2().(2.4.4)

对于固定的 和 λ，以上定理给出了弹性问题令人满意近似的有限元近似。但是这些有限元方法的性能随着 λ 趋向于 而变差。这就是所谓的锁定现象[9]。

令 = (0, 1) × (0, 1). 考虑 = 1 时的纯位移边值问题:

div{2(u λ) + λ tr((u λ)) δ } = fin(2.5.1)

u λ | = 0.

注意给定的 f, 当 λ , divu λ H1() 0. 换句话说，我们正在处理一种几乎不可能压缩的弹性材料。为了强调对 λ 的依赖，将应力张量 σ λ (ν) 和变分形式 a λ (ν , ω) 表示为

σ λ (ν) = 2(ν) + λ tr((ν)) δ ,

a λ (ν , ω) = {2(ν) : (ω) + λ div ν div ω} dx.

(2.5.2)

令 Th为 (图 4) 的一个规则三角剖分。对于每一个 u H2() H1 0(), 定义 u λ h Vh 为以下方程组的特解

a λ (u λ h, ν) = [div σ λ (u)] · ν dx ν Vh, (2.5.3) Vh := { ν H1() : ν |T , T Th}.

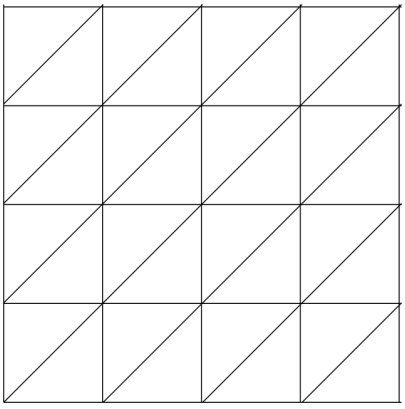
定义 L λ ,h 为

L λ ,h := sup{ |u u λ div σ λ (u)L2() h|H1(): 0 = u H2() H1()}.(2.5.4)

则存在一个与 h 无关的正常数 C 使得[9]

λ inf L λ ,h C. (2.5.5)

式 (2.5.5) 意味着：无论 h 取多小，只要 λ 足够大，我们都能找到 u H2() H1() 使得相对误差 |u uh|H1()/div σ λ (u)L2() 以一个与 h 无关的常数为下界。换句话说，有限元方法的性能将会随着 λ 变大而变坏。



T1

T2

T3

图 4

令 = (0, 1) × (0, 1). 考虑以下纯位移问题

u (+ λ) grad (div u) = f , (2.5.6)

u = 0on ,

其中，f L2(), 1, 2 是 的两个子集，使得 1

, (x, y) 1时 λ = λ1, = 1, (x, y) 2时 λ = λ2, = 2, λ1 = λ2, 1 = 2 = 并且 1 2 =

2.

它的变分形式为，求 u H1() 使得 u| Γ = 0, 并且

as(u, v) = f · v dxdyv H1(), (2.5.7)

其中

as(u, v) = grad u : grad v dxdy + (+ λ)(div u) (div v) dxdy. (2.5.8)

令 Th是 三角划分的一个非退化族。定义

			(2.5.9) (2.5.10) (2.5.11)
(gradh v)|T = grad (v|T),	(divh v)|T = div (v|T),	T ∈ Th.	则问题的离散形式为，求 uh ∈ Vh使得 ah(uh, v) = ∫ f · v dxdy,

11

其中双线性形式 ah(· , ·) 在 V h+ H1() 上的定义为ah(u, v) = gradh u : gradh v dxdy+ (+ λ) (divh u) (divh v) dxdy.定义 Vh+ H1() 上的非协调能量泛函||v||h = ah(v, v)1/2.(2.5.12)

(2.5.13)

显然

||gradh v||L2() ≤ 1/2||v||h.			(2.5.14) (2.5.15) (2.5.16)
定义 (Πhu)(me) = 1 |e| 其中 me 为边缘 e 的中点。则有	∫ e	u ds,	div(Πhu)|T = 1 |T|

并且存在独立于 h 的正常数 C 使得

||u Πhu||L2() + h||gradh (u Πhu)||L2() Ch2|u|2 H(). (2.5.17)

参考文献得

||u||H2() + λ||div u||H1() ≤ C||f||L2(),						,	(2.5.18)
||u uh||h ≤ inf v∈Vh||u v||h +			sup (2.5.20)	|as h(u, v) ||v||h ∫ f · v dxdy|		(2.5.19)	v∈Vh\{0}
∫	gradh u : gradh v dxdy +	∫	△u · v dxdy|	|	∫	≤ C h |u|H2()||gradh v||L2(), divh u divh v dxdy + ∫ grad (div u) · v dxdy|	
(2.5.21)	≤ C h |div u|H1()||gradh v||L2().						(2.5.22)

12

|as h(u, v) f · v dxdy|= |(as h(u, v)|1 f · v dxdy)+ (as h(u, v)|2 f · v dxdy)|(2.5.23)

C h ||gradh v||L2(1)(1|u|H2(1) + (1 + λ1)|div u|H1(1))

+ C h ||gradh v||L2(2)(2|u|H2(2) + (2 + λ2)|div u|H1(2))

C h ||v||h||f||L2().

参考文献得到，存在 u1 H2() H1(), 使得

div u1 = div u, ||u1||H2() ≤ C ||div u||H1(),	(2.5.24) (2.5.25) (2.5.26) (2.5.27)
||u1||H2() ≤ C / 1 + λ ||f||L2(),	divh Πh u1 = divh Πh u.

由公式 (2.5.18), (2.5.17), (2.5.24), (2.5.26) 和 (2.5.27) 得

vVh||u v||h inf

||u Πhu||h

= (1||gradh (u Πhu)||2 L2(1)+ (1 + λ1)||divh (u Πhu)||2 L2(1))1/2

+ (2||gradh (u Πhu)||2 L2(2)+ (2 + λ2)||divh (u Πhu)||2 L2(2))1/2

= (1||gradh (u Πhu)||2 L2(1)+ (1 + λ1)||divh (u1 Πhu1)||2 L2(1))1/2

+ (2||gradh (u Πhu)||2 L2(2)+ (2 + λ2)||divh (u1 Πhu1)||2 L2(2))1/2

C h ||f||L2(). (2.5.28)

由公式 (2.5.19), (2.5.23), (2.5.28) 得

||u uh||h C h ||f||L2(). (2.5.29)

13

3 算例

3.1 弹性问题

3.1.1 算例一

考察以下边值问题

div σ(u) = f

u| = 0. ,

其中 u = (u1, u2)t为求解向量， f = (f1, f2)t为右端向量， = [0, 1] × [0, 1], u1 = (x 1)(y 1)ysin(x),

u2 = (x 1)(y 1)xsin(y).

通过数值实验得到,

1. 当 Lame 常数 = 1, λ = 1 时的误差及误差阶如下表

表 1

h	1.0	0.5	0.25	0.125
|u uh|H1()	4.102804E-1	1.424371E-1	8.382384E-2	4.737414-2
H1 误差阶	1.526284	0.764893	0.823260	0.905945
|u uh|L2()	4.102632E-1	7.121859E-2	2.095596E-2	5.921768E-3
L2 误差阶	2.526284	1.764893	1.823260	1.905945

2. 当 Lame 常数 = 1, λ = 1E4 时的误差及误差阶如下表

表 2

h	1.0	0.5	0.25	0.125
|u uh|H1()	3.954836E-1	1.369558E-1	8.048234E-2	4.543017E-2
H1 误差阶	1.529907	0.7669663	0.8250215	0.9072268
|u uh|L2()	3.945746E-1	6.847791E-2	2.012058E-2	5.678771E-3
L2 误差阶	2.529607	1.766966	1.825021	1.907226

3. 当 Lame 常数 = 1, λ = 1E8 时的误差及误差阶如下表

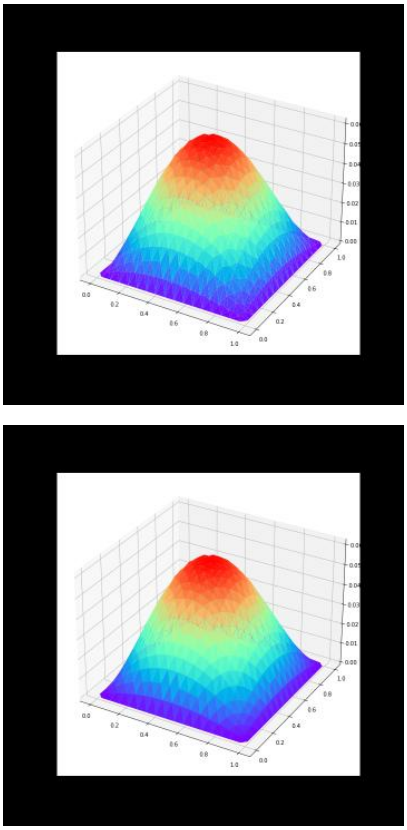
14

表 3

h	1.0	0.5	0.25	0.125
λ	4.102771E-1	1.424359E-1	8.382310E-2	4.737371E-2
|u uh|H1() H1 误差阶	1.526285	0.7648937	0.823261	0.9059459
|u uh|L2() L2 误差阶	4.104650E-1	7.121799E-2	2.095577E-2	5.921714E-3
	2.526285	1.764893	1.823261	1.905945

数值解和精确解图像如下

图 5



(a) 数值解图像(b) 精确解图像

15

3.1.2 算例二

考察以下边值问题

div σ (u) = f

u| = 0. ,

其中 u = (u1, u2)t为求解向量， f = (f1, f2)t为右端向量， = [0, 1] × [0, 1], u1 = x2sin(x 1)y2sin(y 1),

u2 = x2sin(x 1)y2sin(y 1).

通过数值实验得到,

1. 当 Lame 常数 = 1, λ = 1 时的误差及误差阶如下表表 4

h	1.0	0.5	0.25	0.125
|u uh|H1()	1.381670E-1	1.038442E-1	4.190134E-2	2.135923E-2
H1 误差阶	0.4119931	1.309352	0.972136	0.9399994
|u uh|L2()	1.435610E-1	5.192211E-2	1.047533E-2	2.669904E-3
L2 误差阶	1.411993	2.309352	1.972136	1.939999

2. 当 Lame 常数 = 1, λ = 1E4 时的误差及误差阶如下表表 5

h	1.0	0.5	0.25	0.125
|u uh|H1()	1.328102E-1	1.001068E-1	4.029172E-2	2.049834E-2
H1 误差阶	0.4078262	1.312984	0.9749761	0.9414672
|u uh|L2()	1.647602E-1	5.005340E-2	1.007293E-2	2.562293E-3
L2 误差阶	1.407826	2.312984	1.974976	1.941467

3. 当 Lame 常数 = 1, λ = 1E8 时的误差及误差阶如下表

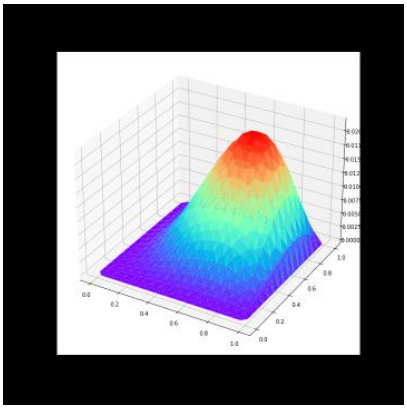
16

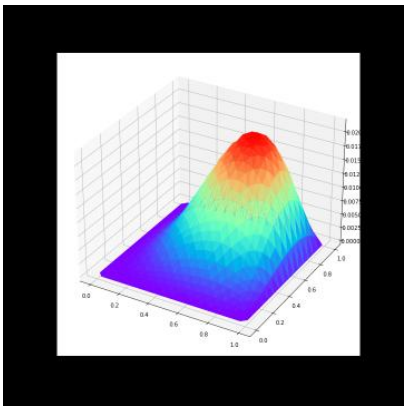
表 6

h	1.0	0.5	0.25	0.125
λ	1.381659E-1	1.038433E-1	4.190099E-2	2.135904E-2
|u uh|H1() H1 误差阶	0.4119922	1.309353	0.9721372	0.9399997
|u uh|L2()	1.473169E-1	5.192169E-2	1.047524E-2	2.669880E-3
L2 误差阶	1.411992	2.309353	1.972137	1.939999

数值解和精确解图像如下

图 6





(a) 数值解图像(b) 精确解图像

17

3.2 带间断系数的弹性问题

3.2.1 算例一

考察以下边值问题

div σ(u) = f,

u| Γ = 0.

其中 u = (u1, u2)t为求解向量，f = (f1, f2)t为右端向量， = [0, 1] ×[0, 1], 1 = [0, 0.5] × [0, 0.5], 2 = [0.5, 1] ×

[0, 0.5], 3 = [0, 0.5] × [0.5, 1], 4 = [0.5, 1] × [0.5, 1].

当 (x, y) 1 时， = λ = λ1,

u1 = x(x 0.5)(x 1)y(y 0.5)(y 1) / λ1,

u2 = x(x 0.5)(x 1)y(y 0.5)(y 1) / λ1.

当 (x, y) 2 时， = λ = λ2,

u1 = x(x 0.5)(x 1)y(y 0.5)(y 1) / λ2,

u2 = x(x 0.5)(x 1)y(y 0.5)(y 1) / λ2.

当 (x, y) 3 时， = λ = λ3,

u1 = x(x 0.5)(x 1)y(y 0.5)(y 1) / λ3,

u2 = x(x 0.5)(x 1)y(y 0.5)(y 1) / λ3.

当 (x, y) 4 时， = λ = λ4,

u1 = x(x 0.5)(x 1)y(y 0.5)(y 1) / λ4,

u2 = x(x 0.5)(x 1)y(y 0.5)(y 1) / λ4.

令 λ = [λ1, λ2, λ3, λ4]， = [1, 2, 3, 4].

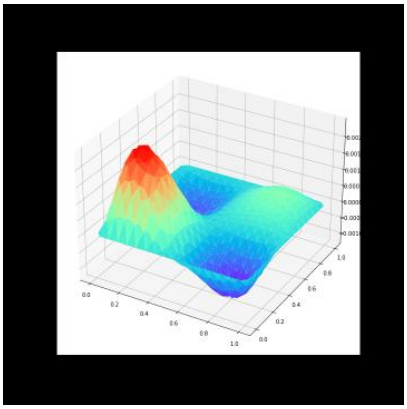
18

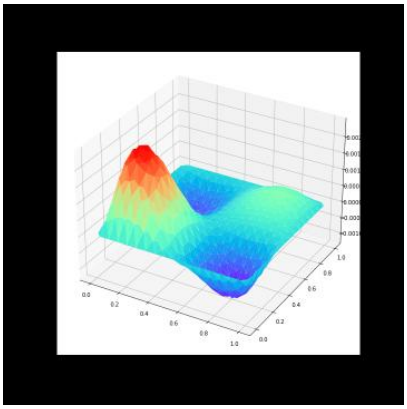
1. 当 Lame 常数 λ = = [1, 2, 3, 4] 时误差如下

表 7

h	0.5	0.25	0.125	0.0625
|u uh|H1()	1.599642E-2	7.130159E-3	3.516517E-3	1.900384E-3
H1 误差阶	1.165743	1.019786	0.8878559	0.9186291
|u uh|L2()	3.999106E-3	8.912698E-4	2.197823E-4	5.938701E-5
L2 误差阶	2.165743	2.019786	1.887855	1.918629

图 7





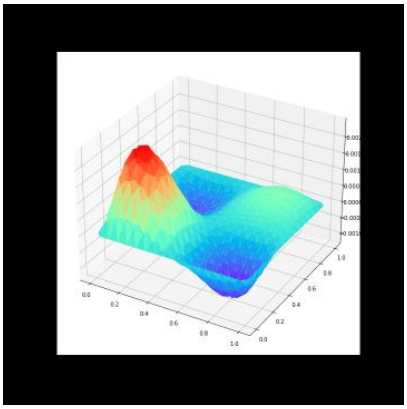
(a) 数值解图像(b) 精确解图像

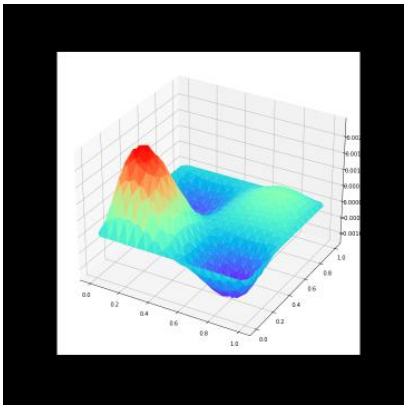
19

2. 当 Lame 常数 λ = = [1E4, 2E4, 3E4, 4E4] 时误差如下表 8

h	0.5	0.25	0.125	0.0625
|u uh|H1()	1.599735E-2	7.130099E-3	3.516522E-3	1.900428E-3
H1 误差阶	1.165839	1.019772	0.8878243	0.918610
|u uh|L2()	3.999338E-3	8.912623E-4	2.197826E-4	5.938839E-5
L2 误差阶	2.165839	2.019772	1.887824	1.91861

图 8





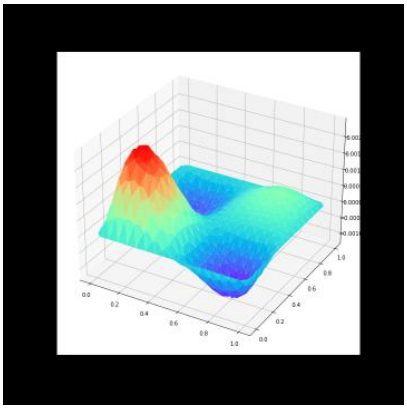
(a) 数值解图像(b) 精确解图像

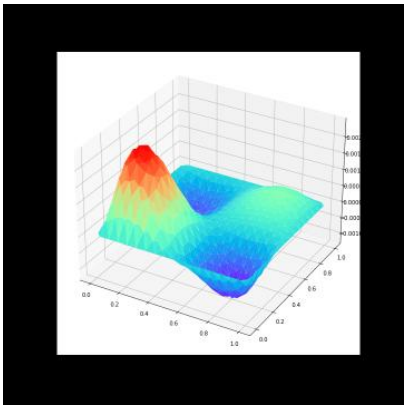
20

3. 当 Lame 常数 λ = = [1E8, 2E8, 3E8, 4E8] 时误差如下表 9

h	0.5	0.25	0.125	0.0625
|u uh|H1()	3.015333E-3	1.139928E-3	6.718610E-4	3.793589E-4
H1 误差阶	1.403373	0.7627090	0.8245995	0.9054011
|u uh|L2()	7.538332E-4	1.424911E-5	4.199131E-5	1.185496E-5
L2 误差阶	2.403373	1.7627090	1.824599	1.905401

图 9





(a) 数值解图像(b) 精确解图像

21

3.2.2 算例二

考察以下边值问题

div σ(u) = f,

u| Γ = 0.

其中 u = (u1, u2)t为求解向量，f = (f1, f2)t为右端向量，= [0, 1] × [0, 1], 1 = [0, 0.5] × [0, 0.5], 2 = [0.5, 1] × [0, 0.5], 3 = [0, 0.5] × [0.5, 1], 4 = [0.5, 1] × [0.5, 1].

当 (x, y) 1 时，= λ = λ1,

u1 = sin(πx) cos(πx) sin(πy) cos(πy) / λ1,

u2 = sin(πx) cos(πx) sin(πy) cos(πy) / λ1.

当 (x, y) 2 时，= λ = λ2,

u1 = sin(πx) cos(πx) sin(πy) cos(πy) / λ2,

u2 = sin(πx) cos(πx) sin(πy) cos(πy) / λ2.

当 (x, y) 3 时，= λ = λ3,

u1 = sin(πx) cos(πx) sin(πy) cos(πy) / λ3,

u2 = sin(πx) cos(πx) sin(πy) cos(πy) / λ3.

当 (x, y) 4 时，= λ = λ4,

u1 = sin(πx) cos(πx) sin(πy) cos(πy) / λ4,

u2 = sin(πx) cos(πx) sin(πy) cos(πy) / λ4.

令 λ = [λ1, λ2, λ3, λ4]，= [1, 2, 3, 4].

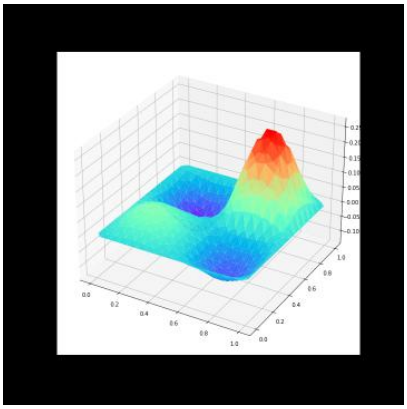
22

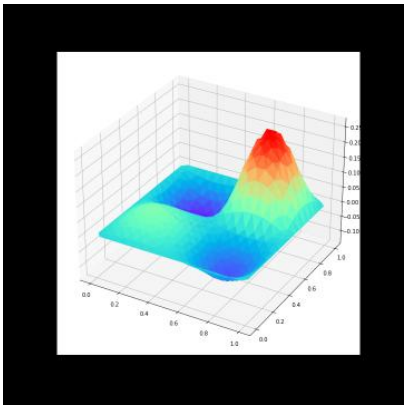
1. 当 Lame 常数 λ = = [4, 3, 2, 1] 时误差如下

表 10: |u uh|H1()

h	0.5	0.25	0.125	0.0625
|u uh|H1()	1.462698	6.166073E-1	4.261624E-1	2.415138E-1
H1 误差阶	1.246208	0.532948	0.819297	0.939899
|u uh|L2()	7.313491E-1	1.541518E-1	5.327030E-2	1.509461E-2
L2 误差阶	2.246208	1.532948	1.819297	1.939899

图 10





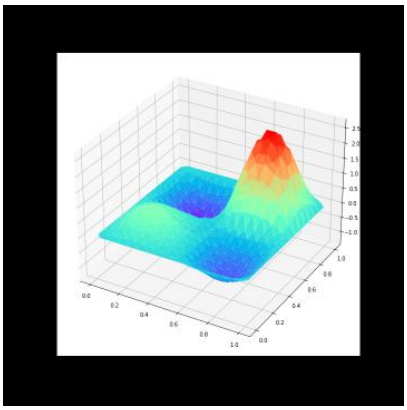
(a) 数值解图像(b) 精确解图像

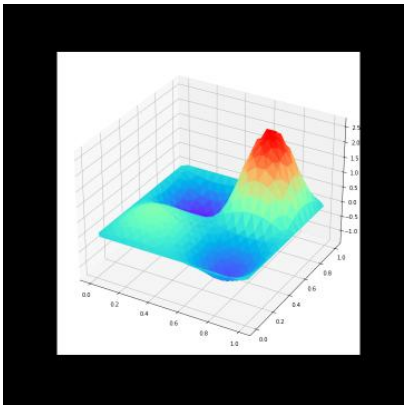
23

2. 当 Lame 常数 λ = = [4E4, 3E4, 2E4, 1E4] 时误差如下表 11

h	0.5	0.25	0.125	0.0625
|u uh|H1()	1.462698E-4	6.166073E-5	4.261624E-5	2.415138E-5
H1 误差阶	1.246208	0.532948	0.819297	0.939899
|u uh|L2()	7.313491E-5	1.541518E-5	5.327030E-6	1.509461E-6
L2 误差阶	2.246208	1.532948	1.819297	1.939899

图 11





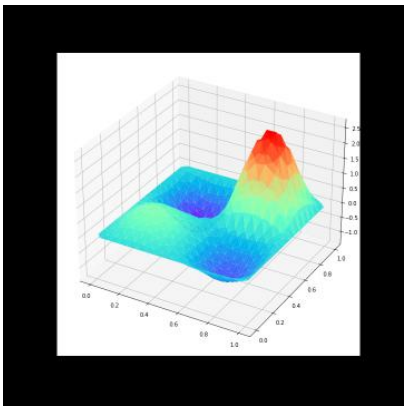
(a) 数值解图像(b) 精确解图像

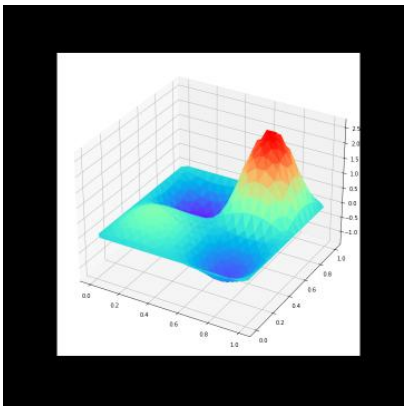
24

3. 当 Lame 常数 λ = = [4E8, 3E8, 2E8, 1E8] 时误差如下表 12: |u uh|H1()

h	0.5	0.25	0.125	0.0625
|u uh|H1()	1.462698E-8	6.166073E-9	2.415138E-9	2.373949E-9
H1 误差阶	1.246208	0.532948	0.819297	0.939899
|u uh|L2()	7.313491E-9	1.541518E-9	5.327030E-10	1.509461E-10
L2 误差阶	2.246208	1.532948	1.819297	1.939899

图 12





(a) 数值解图像(b) 精确解图像

25

4 总结

本文使用 C-R 有限元方法求解了带间断系数的平面弹性问题，分析了非协调元对闭锁现象的影响。数值结果表明，当 Lame 常数间断且相等时，C-R 元可以有效地解除闭锁现象，并且具有预期的收敛阶。

为了完善本文的研究，未来可以考虑对纯牵引力问题进行数值实验，以检验 C-R 元在不同的边界条件下的表现。同时，也可以通过改变间断系数的大小和形式，以及使用不同的网格划分方式，来进一步探究 C-R 元的有效性

和稳定性，以及对间断系数的敏感性。

致谢

时光荏苒，岁月如梭，转眼间大学生活来到了最后阶段. 当我写完这篇毕业论文的时候，有一种如释重负的感觉，感慨颇多. 回首大学四年，得到过太多人的帮助了. 首先诚挚的感谢我的论文指导老师王华老师. 本文的研究工作都是在王华老师的悉心指导下完成的. 王老师平易近人，严谨务实，由于我知识储备不足，在论文撰写过程中遇到了许多困难和疑惑，王华老师都及时给予指点，耐心解释所犯的错误，投入了大量的心血和精力，更是不厌其烦地帮我察看论文中的小漏洞. 王华老师对我的帮忙和关怀实在无法用言语表明. 还要感谢所有的老师们，正是因为有了他们的督促和教导才能让我在这四年的学习生活里受益匪浅，快速汲取专业知识，提升专业能力. 同时也要感谢组内的同学们，是他们以极大的热情来解答我在理论和程序上的疑问，帮忙收集资料，让平淡的日子不再那么枯燥乏味. 最后还要感谢我的家人，是他们的支持与付出才给了我学习的机会，感谢一直对我的理解，这是我不断前进的动力.

附录 A 附录数值程序

Listing 1: elasticityCR.py

```
1
2
3
4
5
6
78
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
2627
28
29
30
31
32
33
```

```
34
35
36
37
  29
38 cm = np.ones(NC, dtype=np.float64) / NC
  39
40cr_node, cr_cell = get_cr_node_cell(node,cell)
  41
42 # 单元刚度矩阵和单元载荷向量
43 A1, A2 = get_stiff_and_div_matrix(cr_node, cr_cell, cm)
44 bb = get_bb(pde, node, cell, cm)
45 cellInOmega = getCellInOmega(cr_node, cr_cell)
46 kk = 1
47 for k in range(4):
48 A1[cellInOmega[k]] *= pde.mu[k]
49A2[cellInOmega[k]] *= pde.mu[k] + pde.lam[k]
50A1, A2, F = get_A1_A2_F(A1, A2, bb, cr_node, cr_cell)
51 A = A1 + A2
  52
```

```
53
54
55
56
57
58
5960
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
```

```
uh = my_solve(A, F, cr_node, getIsBdNode)
u = pde.solution(cr_node, cr_node)
H[j] = np.sqrt(2 * cm[0]) / 2
E[i][j] = H1Error(u, uh)
if j < n-1:
node, cell = uniform_refine(node, cell)
drawer_uh_u(cr_node, uh, u, "../../image/tmp/interface_uh_u/uh_lam ={}.png".
format(Lam[i]), "../../image/tmp/interface_uh_u/u_lam={}.png".
format(Lam[i]))
# 画图 得到误差阶
if n-1 > 1:
# 画图
for i in range( len(Lam)):
fig = plt.figure()
plt.plot(np.log(H[1:]), np.log(E[i][1:]))
plt.title("lam={}". format(Lam[i]))
plt.xlabel("log(h)")
plt.ylabel("log(e)")
plt.savefig(fname="interfaceCRFem/elasticityCRFemLam_{}.png" .
format(Lam[i]))
plt.close(fig)
# 求误差阶
# 得到 P
for i in range( len(Lam)):
f = np.polyfit(np.log(H[1:]), np.log(E[i][1:]) ,1) P[i] = f[0]
```

```
  30
  78
79
80
81
```

Listing 2: tool.py

```
  1
2
```

```
3
4
5
6
7
8
9
101112
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
  31
35 frac_u1_x_y = (2*y-1) * (sin(x) + (x-1) * cos(x))
36 frac_u2_x = 2 * (y-1) * sin(y)
37 frac_u2_y = x * (x-1) * (2*cos(y) - (y-1) * sin(y))
38 frac_u2_x_y = (2*x-1) * (sin(y) + (y-1) * cos(y))
  39
40val[..., 0] = -((2*mu+lam) * frac_u1_x + (mu+lam) * frac_u2_x_y
  + mu*frac_u1_y)
41 val[..., 1] = -((2*mu+lam) * frac_u2_y + (mu+lam) * frac_u1_x_y
  + mu*frac_u2_x)
  42
43 return val
  44
45 def solution(self, p):
46 x = p[..., 0]
47 y = p[..., 1]
```

```
  48
49val = np.zeros(p.shape, dtype=np.float64)
  50
51 val[..., 0] = y * (x - 1) * (y - 1) * np.sin(x)
52 val[..., 1] = x * (x - 1) * (y - 1) * np.sin(y)
  53
54 return val
  55
56 # PDE2
57 # u1 = u2 = x^2 * sin(x-1) * y^2 * sin(y-1)
58 # uh_dir = "../../image/tmp/elaticity_uh_u/PDE2/uh_lam={}.png".format(
  Lam[i])
  59# u_dir = "../../image/tmp/elaticity_uh_u/PDE2/u_lam={}.png".format(Lam[
  i])
60class PDE2():
61 def __init__(self, mu=1, lam=1):
62 self.mu= mu
63 self.lam = lam
64 self.node = np.array([
65 (0,0),
66 (1,0),
67 (1,1),
68 (0,1)], dtype=np.float64)
69self.cell = np.array([(1,2,0), (3,0,2)], dtype=np.int64)
  70
71 def source(self, p):
72 x = p[..., 0]
  32
73 y = p[..., 1]
74 mu= self.mu
75 lam = self.lam
  76
77 sin = np.sin
78 cos = np.cos
79val = np.zeros(p.shape, dtype=np.float64)
  80
81 ux = x**2 * sin(x-1)
82 uy = y**2 * sin(y-1)
83 frac_ux_x = 2 * x * sin(x-1) + x**2 * cos(x-1)
84 frac_ux_xx = 2 * sin(x-1) + 2 * x *cos(x-1) + 2 * x * cos(x-1) -
  x**2 * sin(x-1)
85 frac_uy_y = 2 * y * sin(y-1) + y**2 * cos(y-1)
86 frac_uy_yy = 2 * sin(y-1) + 2 * y *cos(y-1) + 2 * y * cos(y-1) -
  y**2 * sin(y-1)
  87
88 frac_u1_x = frac_ux_xx * uy
```

```
89frac_u1_y = ux * frac_uy_yy
90frac_u1_x_y = frac_ux_x * frac_uy_y
91 frac_u2_x = frac_ux_xx * uy
92 frac_u2_y = ux * frac_uy_yy
93 frac_u2_x_y = frac_ux_x * frac_uy_y
  94
95 val[..., 0] = -((2*mu+lam) * frac_u1_x + (mu+lam) * frac_u2_x_y
  + mu*frac_u1_y)
96 val[..., 1] = -((2*mu+lam) * frac_u2_y + (mu+lam) * frac_u1_x_y
  + mu*frac_u2_x)
  97
98 return val
  99
100def solution(self, p):
101 x = p[..., 0]
102 y = p[..., 1]
  103
104 sin = np.sin
105 val = np.zeros(p.shape, dtype=np.float64)
  106
107 ux = x**2 * sin(x-1)
108 uy = y**2 * sin(y-1)
  109
110val[..., 0] = ux * uy
  33
111 val[..., 1] = ux * uy
  112
113 return val
  114
115 # PDE3
116 # u1 = u2 = -(x-1) * (e^x-1) * (y-1) * (e^y-1)
117 # uh_dir = "../../image/tmp/elaticity_uh_u/PDE3/uh_lam={}.png".format(
  Lam[i])
  118# u_dir = "../../image/tmp/elaticity_uh_u/PDE3/u_lam={}.png".format(Lam[
  i])
119class PDE3():
120def __init__(self, mu=1, lam=1):
121 self.mu= mu
122 self.lam = lam
123 self.node = np.array([
124 (0,0),
125 (1,0),
126 (1,1),
127 (0,1)], dtype=np.float64)
128 self.cell = np.array([(1,2,0), (3,0,2)], dtype=np.int64)
  129
```

```
130def source(self, p):
131 x = p[..., 0]
132 y = p[..., 1]
133 mu= self.mu
134 lam = self.lam
 135
136 sin = np.sin
137 cos = np.cos
138 exp = np.exp
139val = np.zeros(p.shape, dtype=np.float64)
 140
141 ux = (x-1) * (exp(x) - 1)
142 uy = (y-1) * (exp(y) - 1)
143 frac_ux_x = x * exp(x) - 1
144 frac_ux_xx = exp(x) * (x+1)
145 frac_uy_y = y * exp(y) - 1
146 frac_uy_yy = exp(y) * (y+1)
 147
148 frac_u1_x = frac_ux_xx * uy
149frac_u1_y = ux * frac_uy_yy
150frac_u1_x_y = frac_ux_x * frac_uy_y
 34
151 frac_u2_x = frac_ux_xx * uy
152 frac_u2_y = ux * frac_uy_yy
153 frac_u2_x_y = frac_ux_x * frac_uy_y
 154
155 val[..., 0] = -((2*mu+lam) * frac_u1_x + (mu+lam) * frac_u2_x_y
 + mu*frac_u1_y)
156 val[..., 1] = -((2*mu+lam) * frac_u2_y + (mu+lam) * frac_u1_x_y
 + mu*frac_u2_x)
 157
158 return -val
 159
160def solution(self, p):
161 x = p[..., 0]
162 y = p[..., 1]
 163
164 val = np.zeros(p.shape, dtype=np.float64)
 165
166 ux = (x-1) * (np.exp(x) - 1)
167 uy = (y-1) * (np.exp(y) - 1)
 168
169val[..., 0] = ux * uy
170val[..., 1] = ux * uy
 171
172 return -val
```

```
173
174 # interfaceData1
175 # u1 = u2 = x(x-0.5)(x-1) * y(y-0.5)(y-1)
176 # uh_dir = "../../image/tmp/interface_uh_u/PDE1/uh_lam={}.png".format(
  Lam[i])
  177# u_dir = "../../image/tmp/interface_uh_u/PDE1/u_lam={}.png".format(Lam[
  i])
178
179
180
181
182
183
184
185
186
187
188class interfaceData():
def __init__(self, mu=np.array([1,2,3,4]), lam=np.array([1,2,3,4])): self.mu = mu
self.lam = lam
self.node = np.array([(0,0),
(0.5,0),
(1,0),
(0,0.5),
(0.5,0.5),
(1,0.5),
(0,1),
  35
189(0.5,1),
190(1,1)], dtype=np.float64)
191 self.cell = np.array([[0,1,4],
192 [0,4,3],
193 [1,2,5],
194 [1,5,4],
195 [3,4,7],
196 [3,7,6],
197 [4,5,8],
198 [4,8,7]], dtype=np.int64)
  199
200def source(self, p):
201 x = p[..., 0]
202 y = p[..., 1]
  203
204 mu= 1
205 lam = 1
206 val = np.zeros(p.shape, dtype=np.float64)
```

```
  207
208 frac_u1_x = (6*x-3) * (y**3-(3/2)*y**2+(1/2)*y)
209frac_u1_y = (x**3-(3/2)*x**2+(1/2)*x) * (6*y-3)
210frac_u1_x_y = (3*x**2-3*x+1/2) * (3*y**2-3*y+1/2)
211 frac_u2_x = frac_u1_x
212 frac_u2_y = frac_u1_y
213 frac_u2_x_y = frac_u1_x_y
  214
215 val[..., 0] = -((2*mu+lam) * frac_u1_x + (mu+lam) * frac_u2_x_y
  + mu*frac_u1_y)
216 val[..., 1] = -((2*mu+lam) * frac_u2_y + (mu+lam) * frac_u1_x_y
  + mu*frac_u2_x)
  217
218 return val
  219
220def solution(self, p, cr_node):
221 x = p[..., 0]
222 y = p[..., 1]
  223
224 val = np.zeros(p.shape, dtype=np.float64)
  225
226 val[..., 0] = x * (x - 0.5) * (x - 1) * y * (y - 0.5) * (y - 1)
227 val[..., 1] = x * (x - 0.5) * (x - 1) * y * (y - 0.5) * (y - 1)
  228
  36
229crCell = getWhichCell(cr_node)
230for i in range(4):
231 val[crCell[i], :] /= self.lam[i]
  232
233 return val
  234
235 def H1Error(u, uh):
236 tmp = u - uh
237 e = np.einsum("ni, ni -> n", tmp, tmp)
238 sum = e. sum()
239return np.sqrt( sum)
  240
```

241
242
243244
245
246
247
248
249
250
	def print_error(Lam, H, E):
	for i in range(len(Lam)):
	print("----------Lam= {}----------". format(Lam[i]))
	n = H.shape[0]
	print()
	for j in range(n):
	print("h= ", H[j])
	print("e=", E[i][j])
	print()
	print()

```
251
252 def print_P(Lam, P):
253 print("----------误差阶----------")
254 for i in range( len(Lam)):
255 print("lam= ", Lam[i])
256 print("p= ", P[i])
257 print()
258
259#判断 P (维度[2]) 是否在 cr_node, 是则放回其下标， 否则 (val = 1 时) 将 P
加入 cr_node 并返回下标
```

260
261
262
263
264
265
266267
268
	def is_in_cr_node(p, cr_node):
	#p 不会为[0,0]
	index = np.where((cr_node == p). all(axis=1))[0]
	if len(index):
	return index[0]
	else:
	in_index = np.where((cr_node == np.array([0,0])).
	all(axis=1))[0]
	if len(in_index) == 0:
	print("cr_node= ", cr_node)

```
37
269raise Exception("数组cr_node已满")
270cr_node[in_index[0]] = p
271 return in_index[0]
272
273 #判断 P (维度[2]) 是否在 node, 是则放回其下标， 否则将 P 加入 node 并返回
下标
274 def is_in_node(p, node):
275 #p 不会为[0,0]
276 index = np.where((node == p). all(axis=1))[0]
277 if len(index):
278 return index[0]
279else:
280in_index = np.where((node == np.array([0,0])). all(axis=1))[0]
281 if len(in_index) == 1:
282 print("node= ", node)
283 raise Exception("数组node已满")
284 node[in_index[1]] = p
285 return in_index[1]
286
287 # a_cell 是否属于 cell， 是则返回下标， 否则返回 -1
288 def is_in_cell(a_cell, cell):
289i = np.where((cell == a_cell). all(axis=1))[0]
290if len(i):
291 return i[0]
292 else:
293 return -1
294
```

```
295
296
297298
299
300
301#将 a_cell (维数[3]) 放入new_cr_cell
def push_cr_cell(a_cell, new_cr_cell):
in_index = np.where((new_cr_cell == np.array([0,0,0])).
all(axis=1))[0]
if len(in_index) == 0:
raise Exception("数组cr_cell已满") new_cr_cell[in_index[0]] = a_cell
return new_cr_cell
  302
  303
304 # 对单个三角形 a_cell_node (维度 [3, 2]) 求三条边中点 p1, p2, p3 并将其
  放入 new_cr_node 、 new_cr_cell
  305
306
307
308def a_creat(a_cell_node, new_cr_node, new_cr_cell): p1 = (a_cell_node[0] + a_cell_node[1]) / 2 p2 = (a_cell_node[0] +
a_cell_node[2]) / 2 p3 = (a_cell_node[1] + a_cell_node[2]) / 2
  38
  309
310p1_i = is_in_cr_node(p1, new_cr_node)
311 p2_i = is_in_cr_node(p2, new_cr_node)
312 p3_i = is_in_cr_node(p3, new_cr_node)
  313
314 push_cr_cell([p1_i, p2_i, p3_i], new_cr_cell)
  315
316 return new_cr_node, new_cr_cell
  317
318 def refine_a_cell(a_cell, new_node, new_cell):
319p1 = (new_node[a_cell][0] + new_node[a_cell][1]) / 2
320p2 = (new_node[a_cell][0] + new_node[a_cell][2]) / 2
321 p3 = (new_node[a_cell][1] + new_node[a_cell][2]) / 2
  322
323 p1_i = is_in_node(p1, new_node)
324 p2_i = is_in_node(p2, new_node)
325 p3_i = is_in_node(p3, new_node)
  326
327 push_cr_cell([p1_i, p2_i, p3_i], new_cell)
328 push_cr_cell([a_cell[0], p1_i, p2_i], new_cell)
329push_cr_cell([p1_i, a_cell[1], p3_i], new_cell)
330push_cr_cell([p2_i, p3_i, a_cell[2]], new_cell)
  331
332 return new_node, new_cell
```

```
333
334 # 从剖分node, cell得到 cr_node, cr_cell
335 # 单元数 NC
336 # 剖分次数 n : log_4(NC / 2)
337 # 外边 out_edge : 4 * 2**n
338 # 总边 all_edge ： 3 * NC - (3 * NC - out_edge) / 2
339def get_cr_node_cell(node, cell):
340NC = cell.shape[0]
341 # n 特定情况下剖分次数
342 n = math.log(NC/2, 4)
343 NN = int(3 * 2 * 4**n - (3 * 2 * 4**n - 4 * 2**n) / 2)
344 cr_node = np.zeros((NN, 2), dtype=np.float64)
345 cr_cell = np.zeros_like(cell)
346
347 for i in range(NC):
348 cr_node, cr_cell = a_creat(node[cell[i]], cr_node, cr_cell)
349
350return cr_node, cr_cell
39
351
352 # 返回 node 中是否为边界点的信息
353 # isBdNode [NN] bool
354 def getIsBdNode(cr_node):
355 is_BdNode = np.zeros(cr_node.shape[0], dtype= bool)
356 for i in range(cr_node.shape[0]):
357 a = np. min(np. abs(cr_node[i] - np.array([0,0])))
358 b = np. min(np. abs(cr_node[i] - np.array([1,1])))
359if a 〈 1e-13 or b 〈 1e-13:
360is_BdNode[i] = True
361 return is_BdNode
362
363 def getIsBdLineNode(cr_node):
364 NN = cr_node.shape[0]
365 isBdLineNode = getIsBdNode(cr_node)
366 for i in range(NN):
367 a = cr_node[i,0]
368 b = cr_node[i,1]
369if a == 0.5 or b == 0.5:
370isBdLineNode[i] = True
371 return isBdLineNode
372
373 def uniform_refine(node, cell):
374 old_NN = node.shape[0]
375 old_NC = cell.shape[0]
376 n = math.log(old_NC/2, 4)
377 NC = 4 * old_NC
```

```
num_edge = int(3 * 2 * 4**n - (3 * 2 * 4**n - 4 * 2**n) / 2)
NN = old_NN + num_edge

new_node = np.zeros((NN, 2), dtype=np.float64)
new_cell = np.zeros((NC, 3), dtype=np.int64)
new_node[:old_NN] = node

for i in range(old_NC):
new_node, new_cell = refine_a_cell(cell[i], new_node, new_cell)

return new_node, new_cell

def get_cr_glam_and_pre(cr_node, cr_cell):
NC = cr_cell.shape[0]
NN = cr_node.shape[0]

cr_node_cell = cr_node[cr_cell]
##求解CR元导数
cr_node_cell_A = np.ones((NC, 3, 3), dtype=np.float64)
#求解CR元导数的系数矩阵
cr_node_cell_A[:, :, 0:2] = cr_node_cell
#用于求解CR元的值
# cr_glam_x_y_pre [NC, 3, 3]
cr_glam_x_y_pre = np.zeros((NC, 3, 3), dtype=np.float64)
for k in range(NC):
cr_glam_x_y_pre[k, :, :] = solve(cr_node_cell_A[k, :, :], np.
diag(np.ones(3)))
#[NC,3,3]
cr_glam_x_y = np.copy(cr_glam_x_y_pre)
cr_glam_x_y = cr_glam_x_y[:, 0:2, :]
cr_glam_x_y = cr_glam_x_y.transpose((0,2,1))
return cr_glam_x_y, cr_glam_x_y_pre

# phi_val [NC,3(点),6(6个基函数),2(两个分量)]
def get_phi_val(node, cell, cr_glam_pre):
NC = cell.shape[0]
# cr_node_val [NC,3(点),3(三个cr元的值)] CR元在各顶点的值
node_cell_A = np.ones((NC,3,3), dtype=np.float64)
node_cell_A[:,:,0:2] = node[cell]
cr_node_val = np.einsum("cij, cjk -> cik", node_cell_A, cr_glam_pre)

# phi_node_val [NC,3(点),6(6个基函数),2(两个分量)]
phi_node_val = np.zeros((NC,3,6,2), dtype=np.float64)
phi_node_val[:,:,0:5:2,0] = cr_node_val
phi_node_val[:,:,1:6:2,1] = cr_node_val
return phi_node_val
```

```
422
423 def get_phi_grad_and_div(cr_node, cr_cell):
424 NC = cr_cell.shape[0]
425 cr_glam_x_y, cr_glam_x_y_pre = get_cr_glam_and_pre(cr_node, cr_cell)
426 #求 cr_phi_grad [NC,6(基函数),2(分量 x , y),2(导数)]
427 cr_phi_grad = np.zeros((NC,6,2,2), dtype=np.float64)
428 cr_phi_grad[:, 0:5:2, 0, :] = cr_glam_x_y
429cr_phi_grad[:, 1:6:2, 1, :] = cr_glam_x_y
430
431 # cr_phi_div [NC, 6]
432 #cr_phi_div = np.einsum("cmij -> cm", cr_phi_grad)
433 cr_phi_div = cr_glam_x_y.copy()
41
434 cr_phi_div = cr_phi_div.reshape(NC, 6)
435 return cr_phi_grad, cr_phi_div
436
437 ## 单元刚度矩阵， 单元质量矩阵 stiff, div [NC, 6, 6]
438 def get_stiff_and_div_matrix(cr_node, cr_cell, cm):
439cr_phi_grad, cr_phi_div = get_phi_grad_and_div(cr_node, cr_cell)
440
441 ## 单元刚度矩阵
442 # A1 A2 [NC, 6, 6]
443 A1 = np.einsum("cnij, cmij, c -> cnm", cr_phi_grad, cr_phi_grad, cm)
444 A2 = np.einsum("cn, cm, c -> cnm", cr_phi_div, cr_phi_div, cm)
445 return A1, A2
446
447 ## 单元载荷向量 bb [NC, 6]
448 def get_bb(pde, node, cell, cm):
449cr_node, cr_cell = get_cr_node_cell(node,cell)
450cr_glam_x_y, cr_glam_x_y_pre = get_cr_glam_and_pre(cr_node, cr_cell)
451 # phi_val [NC,3(点),6(6个基函数),2(两个分量)]
452 phi_node_val = get_phi_val(node, cell, cr_glam_x_y_pre)
453
454 # val [NC,3(点),2(分量)] 右端项在各顶点的值
455 val = pde.source(node[cell])
456
457 # phi_val [NC,3,6] 基函数和右端项的点乘
458 phi_val = np.einsum("cijk, cik -> cij", phi_node_val, val)
459# bb [NC,6]
460bb = phi_val. sum(axis=1) * cm[0] / 3
461 return bb
462
463 #input
464 #单元刚度矩阵 A1, A2 [NC, 6, 6], bb [NC,6]
465 # output
466 # 总刚度矩阵 A1, A2 [2*NN,2*NN], F [2*NN]
```

```
467 def get_A1_A2_F(A1, A2, bb, cr_node, cr_cell):
468 NN = cr_node.shape[0]
469NC = cr_cell.shape[0]
470# cell_x_y [NC, 3(三个点), 2(x y 方向上基函数的编号)]
471 cell_x_y = np.broadcast_to(cr_cell[:,:,None], shape=(NC, 3, 2)).copy
  ()
  472cell_x_y[:,:,0] = 2 * cell_x_y[:,:,0]#[NC,3] 三个节点x方向上
  基函数在总刚度矩阵的位置
  473cell_x_y[:,:,1] = 2 * cell_x_y[:,:,1] + 1#[NC,3] 三个节点y方向上
  42
  基函数在总刚度矩阵的位置
474 cell_x_y = cell_x_y.reshape(NC, 6)
475 I = np.broadcast_to(cell_x_y[:, :, None], shape=A1.shape)
476 J = np.broadcast_to(cell_x_y[:, None, :], shape=A2.shape)
  477
478 A1 = csr_matrix((A1.flat, (I.flat, J.flat)), shape=(2 * NN,2 * NN))
479A2 = csr_matrix((A2.flat, (I.flat, J.flat)), shape=(2 * NN,2 * NN))
480F = np.zeros(2 * NN)
481 np.add.at(F, cell_x_y, bb)
482 return A1, A2, F
  483
484 def my_solve(A, F, cr_node, getIsBdNode):
485 NN = cr_node.shape[0]
486 isBdNode = getIsBdNode(cr_node)
487 isInterNode = ~isBdNode
488 #print("isInterNode= ", isInterNode)
489isInterNodeA = np.broadcast_to(isInterNode[:, None], shape=(NN, 2))
490isInterNodeA = isInterNodeA.reshape(2 * NN)
491 #print("isInterNodeA= ", isInterNodeA)
  492
493 uh = np.zeros((2 * NN), dtype=np.float64)
494 uh[isInterNodeA] = spsolve(A[:, isInterNodeA][isInterNodeA], F[
  isInterNodeA])
495 #uh = spsolve(A, F)
496 #print("uh= ", uh)
497 uh = uh.reshape(NN, 2)
498 return uh
  499
500## [4,NN] 返回各点属于哪个区间
501 ## \Omega_1 [0,0.5] \times [0,0.5)
502 ## \Omega_2 (0.5, 1] \times [0,0.5]
503 ## \Omega_3 [0,0.5) \times [0.5,1]
504 ## \Omega_4 [0.5,1] \times (0.5,1]
505 def getWhichCell(node):
506 isWhichCellNode = np.zeros((4,node.shape[0]), dtype= bool)
507 for i in range(node.shape[0]):
```

```
a = node[i, 0] - 0
b = node[i, 1] - 0
if a <= 0.5 and b < 0.5:
isWhichCellNode[0,i] = True
if a > 0.5 and b <= 0.5:
isWhichCellNode[1,i] = True
43
if a < 0.5 and b >= 0.5:
isWhichCellNode[2,i] = True
if a >= 0.5 and b > 0.5:
isWhichCellNode[3,i] = True
return isWhichCellNode
519
## [4, NC] 返回各单元属于哪个区间
def getCellInOmega(node, cell):
cellInOmega = np.zeros(cell.shape[0], dtype= bool)
#print("node[cell]= ", node[cell])
mid_p = node[cell]. sum(axis=1) / 3
#print("mid_p= ", mid_p)
cellInOmega = getWhichCell(mid_p)
return cellInOmega
528
## [4,NN]
## \line_1 x=0.5, y<0.5
## \line_2 x=0.5, y> 0.5
## \line_3 x<0.5, y=0.5
## \line_4 x> 0.5, y=0.5
def getInterfaceCell(node):
interfaceCell = np.zeros((4,node.shape[0]), dtype= bool)
for i in range(node.shape[0]):
a = node[i,0]
b = node[i,1]
if a == 0.5 and b < 0.5:
interfaceCell[0,i] = True
if a == 0.5 and b > 0.5:
interfaceCell[1,i] = True
if a < 0.5 and b == 0.5:
interfaceCell[2,i] = True
if a > 0.5 and b == 0.5:
interfaceCell[3,i] = True
return interfaceCell
548
def getInterLineNode(node):
NN = node.shape[0]
lineNode = np.zeros(NN, dtype= bool)
for i in range(NN):
```

```
553 a = node[i,0]
554 b = node[i,1]
555 if a == 0.5 or b == 0.5:
  44
  556
557
558
559
560561
562
563
564565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
  45
```


参考文献

[1] 王兆清，徐子康，李金. 不可压缩平面问题的位移-压力混合重心插值配点法. 应用力学学报, 35(3):631636, 2018.

[2] 陈绍春，肖留超. 平面弹性的一个新的 locking-free 非协调有限元. 应用数学, 20(4):739747, 2007.

[3] YT Peet and PF Fischer. Legendre spectral element method with nearly incompressible materials. European Journal of Mechanics-A/Solids, 44:91103, 2014.

[4] Arif Masud, Timothy J Truster, and Lawrence A Bergman. A variational multiscale a posteriori error estimation method for mixed form of nearly incompressible elasticity. Computer Methods in Applied Mechanics and En-gineering, 200(47-48):34533481, 2011.

[5] Ferdinando Auricchio, L Beirao Da Veiga, Carlo Lovadina, and Alessandro Reali. An analysis of some mixed-enhanced finite element for plane linear elasticity. Computer Methods in Applied Mechanics and Engineering, 194(27-29):29472968, 2005.

[6] Peter Hansbo and Mats G Larson. Discontinuous galerkin and the crouzeixraviart element: application to elasticity. ESAIM: Mathematical Modelling and Numerical Analysis, 37(1):6372, 2003.

[7] 李荣华，刘播. 偏微分方程数值解, 2007.

[8] 陈纪修，於崇华，金路. 数学分析. 高等教育出版社, 2004.

[9] Susanne C Brenner, L Ridgway Scott, and L Ridgway Scott. The mathe-matical theory of finite element methods, volume 3. Springer, 2008.

指标说明

1、**总相似比：**类似于重合率，即送检论文内容与所选检测资源范围内所有文献相似的部分（包括参考引用部分），占整个送检论文内容的比重，总相似比=复写率+引用率；

2、**复写率：**即送检论文内容与所选检测资源范围内所有文献相似的部分（不包括参考引用部分），占整个送检论文内容的比重；

3、**引用率：**即送检论文内容中被系统识别为引用的部分，占整个送检论文内容的比重（引用部分一般指正确标示引用的部分）；

4、**自写率：**即送检论文内容中剔除相似片段和引用片段后，占整个送检论文内容的比重，一般可用于论文的原创性和新颖性评价，自写率=1-复写率-引用率；

5、**同届相似比：**即送检论文内容与校方所选同届库检测资源范围内所有文章相似的部分（不包括参考引用部分），占整个送检论文内容的比重；

6、报告中，红色与橙色文字表示复写片段，浅蓝色与蓝色、深蓝色文字表示引用片段，紫色文字表示同届相似片段，黑色文字表示自写片段。

免责声明

1、 本报告为G · 格子达系统检测后自动生成，鉴于论文检测技术及论文检测样本库的局限性，G · 格子达不保证检测报告的绝对准确，您所选择的检测资源范围内的检验结果及相关结论仅供参考，不得作为其他任何依据；

2、 G · 格子达论文检测服务中使用的论文样本，除特别声明者外，其著作权归各自权利人享有。根据《中华人民共和国著作权法》等相关法律法规，G · 格子达网站仅为学习研究、介绍、科研等目的引用论文片段。除非经原作者许可，请勿超出合理使用范围使用本网站提供的检测报告及其他内容。

联系我们

防伪二维码

关注微信公众号

官方网站:co.gocheck.cn

客服热线:400-699-3389

客服QQ:800113999