

分类号 O342
密 级 公开

单位代码 10618
学 号 622200990040



重庆交通大学

专业硕士学位论文

ABAQUS-MATLAB 拓扑优化集成方 法及碰撞拓扑优化研究

研 究 生 姓 名: 靳立涵
导师姓名及职称: 朱孙科 副教授

申请专业学位类别 机械硕士 学位授予单位 重庆交通大学
论文提交日期 2023 年 4 月 7 日
专 业 领 域 名 称 车辆工程 论文答辩日期 2023 年 5 月 28 日

2023 年 6 月 6 日

Research on ABAQUS-MATLAB Topology Optimization Integration Method and Crash Topology Optimization

A Dissertation Submitted for the Degree of Master

Candidate: Jin Lihan

Supervisor: Prof. Zhu Sunke

Chongqing Jiaotong University, Chongqing, China

摘 要

拓扑优化是一种根据载荷工况、约束条件和性能指标,在设计区域内进行材料分布优化的方法,是产品概念设计的重要方法之一。目前,众多拓扑优化方法实现的基础程序主要采用 MATLAB 语言编写,其往往只能进行简单典型算例的拓扑优化,存在难以直接用于复杂系统拓扑优化设计的问题,而以 ABAQUS 为代表的商业软件具有强大的有限元计算能力,在接触碰撞分析方面尤为突出。开展 ABAQUS-MATLAB 拓扑优化集成框架及其平台研究,为拓扑优化基础程序实现复杂系统的快速应用提供有力支撑,具有重要的理论意义和实际应用价值。

本文提出了一种针对模型数据库文件的 ABAQUS-MATLAB 拓扑优化集成框架,该框架不同于采用第三方工具包和输入文本文件类型的集成方法,其扩展性强、可视化程度高、稳定性好;而后,实现了基于固体各向同性材料惩罚法(SIMP)的 ABAQUS-MATLAB 拓扑优化集成方法,通过多个典型数值算例验证了该集成方法的可行性;进而,结合双向渐进结构拓扑优化法(BESO)和等效静态载荷法(ESLO)实现了碰撞拓扑优化集成方法,获得了以刚度最大化为目标的汽车吸能盒碰撞拓扑优化结果;最后,发展了基于该集成框架的混合元胞自动机(HCA)碰撞拓扑优化平台,开展了汽车吸能盒耐撞性拓扑优化研究。论文的主要研究内容如下:

(1) 分析了 ABAQUS 文件类型与脚本接口、MATLAB 语言特点,确定了基于模型数据库文件的 ABAQUS-MATLAB 拓扑优化集成框架总体方案,阐述了 SIMP 法的拓扑优化原理及其与所提出集成框架的关系,为后续拓扑优化集成方法实现提供理论和条件支撑。

(2) 研究了采用 SIMP 法建立结构拓扑优化集成方法所需的 Python 程序、MATLAB 程序及其数据传递关系,建立了基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成方法,利用该方法在 ABAQUS-MATLAB 平台中对二维简支梁、三维 L 型异型梁、多应力工况悬臂梁、含孔洞悬臂梁和轮毂装配结构等多个典型算例开展拓扑优化,并与相同条件下 SIMP 法的基础代码拓扑优化结果进行了对比,验证了该拓扑优化集成方法的有效性。

(3) 实现了基于 BESO 法和等效静态载荷法的碰撞拓扑优化集成方法,并对冲击载荷下固支梁和悬臂梁算例进行了动态拓扑优化,将该优化结果与已有 BESO 法和 ESLO 法的冲击拓扑优化结果做了比较,两者的拓扑构型基本一致。采用该碰撞拓扑优化集成方法,开展了冲击载荷下 AL/CFRP 汽车吸能盒刚度最大化的动态拓扑优化。

(4) 实现了基于混合元胞自动机法的耐撞性拓扑优化集成方法, 通过与典型算例拓扑优化结果的对比, 表明了使用集成方法后耐撞性拓扑优化法在 ABAQUS-MATLAB 平台运行的可行性。分析了不同更新规则和自动机邻域形状对拓扑优化结果的影响, 以提高耐撞性为目标对汽车吸能盒进行了碰撞拓扑优化研究。

关键词: 拓扑优化, ABAQUS, MATLAB, 接触碰撞, 汽车吸能盒

Abstract

Topology optimization is a method to optimize material distribution in the design area according to load conditions, constraint conditions and performance indicators, and is one of the important methods of conceptual product design. At present, many topology optimization methods are mainly written in MATLAB language, which can only optimize the topology of simple and typical cases and is difficult to be directly used in the topology optimization design of complex systems. However, commercial software represented by ABAQUS has strong finite element computing ability, especially in contact collision analysis. It is of great theoretical significance and practical application value to carry out the research of ABAQUS-MATLAB topology optimization integration framework and its platform to provide strong support for the topology optimization basic program to realize the rapid application of complex systems.

This thesis proposes an ABAQUS-MATLAB topology optimization integration framework for model database files, which is different from the integration methods using third-party toolkits and input text file types, and is highly scalable, visualized, and stable; then, an ABAQUS-MATLAB topology optimization integration method based on the solid isotropic material penalty method (SIMP) is implemented. Then, the integrated ABAQUS-MATLAB topology optimization method based on the solid isotropic material penalty method (SIMP) is implemented, and the feasibility of the integrated method is verified by several typical numerical cases. Finally, a hybrid cellular automaton (HCA) crash topology optimization platform based on this integrated framework is developed to carry out the research on the topology optimization of the crashworthiness of the energy-absorbing box of an automobile. The main research contents of the paper are as follows:

(1) Analyzed the ABAQUS file type and script interface, MATLAB language features, determined the overall scheme of ABAQUS-MATLAB topology optimization integration framework based on the model database file, elaborated the topology optimization principle of SIMP method and its relationship with the proposed integration framework, and provided theoretical and conditional support for the subsequent topology optimization integration method implementation.

(2) The Python program, MATLAB program and its data transfer relationship required to establish the structural topology optimization integration method by SIMP method are studied, and the ABAQUS-MATLAB topology optimization integration method based on SIMP method is established, and the method is used to carry out topology optimization in the ABAQUS-MATLAB platform for two-dimensional simply supported beam, three-dimensional L-shaped shaped beam, multi-stress condition The topology optimization method is compared with the topology optimization results of SIMP method under the same conditions, and the effectiveness of the topology optimization integration method is verified.

(3) The integrated crash topology optimization method based on the BESO method and the equivalent static load method is implemented, and the dynamic topology optimization of the solid-supported beam and cantilever beam cases under impact loading

is carried out. The dynamic topology optimization for maximizing the stiffness of AL/CFRP automotive energy-absorbing box under impact loading is carried out using this integrated collision topology optimization method.

(4) An integrated method of crashworthiness topology optimization based on the hybrid cellular automata method is implemented, and the feasibility of the crashworthiness topology optimization method running on the ABAQUS-MATLAB platform after using the integrated method is demonstrated by comparing the topology optimization results with those of typical algorithms. The effects of different update rules and automata neighborhood shapes on the topology optimization results are analyzed, and a crash topology optimization study is carried out for the car energy-absorbing box with the goal of improving the crashworthiness.

KEY WORDS: Topology optimization, ABAQUS, MATLAB, Contact crash, Car energy absorption box

目 录

第一章 绪论.....	1
1.1 课题研究意义.....	1
1.2 国内外研究现状及趋势.....	2
1.2.1 拓扑优化方法及基础程序.....	2
1.2.2 碰撞拓扑优化方法.....	5
1.3 ABAQUS 和 MATLAB 软件简介.....	6
1.4 本文研究工作.....	7
第二章 ABAQUS-MATLAB 拓扑优化集成框架设计方案.....	9
2.1 引言.....	9
2.2 SIMP 法拓扑优化理论.....	9
2.2.1 SIMP 法拓扑优化模型.....	9
2.2.2 优化规则.....	10
2.2.3 灵敏度过滤方案.....	11
2.3 ABAQUS-MATLAB 拓扑优化集成框架设计方案.....	11
2.3.1 ABAQUS 文件类型与脚本接口.....	12
2.3.2 集成框架中 MATLAB 程序结构.....	14
2.4 本章小结.....	15
第三章 基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成方法.....	17
3.1 引言.....	17
3.2 MATLAB 核心程序及解读.....	17
3.2.1 初始化部分.....	17
3.2.2 循环控制部分.....	18
3.2.3 设计变量更新子函数.....	19
3.2.4 网格过滤子函数.....	20
3.3 Python 核心程序及解读.....	21
3.3.1 有限元分析初始化.....	22
3.3.2 设计变量组装子函数.....	22
3.3.3 有限元模型载入.....	23
3.3.4 材料属性更新.....	24
3.3.5 有限元后处理.....	24
3.4 拓扑优化算例及功能扩展.....	25
3.4.1 二维和三维算例.....	25
3.4.2 数据交互扩展.....	27
3.4.3 多载荷工况扩展.....	27

3.4.4 装配体结构扩展.....	28
3.5 本章小结.....	29
第四章 基于等效静态载荷法的拓扑优化集成方法及应用	31
4.1 引言.....	31
4.2 等效静态载荷法与 BESO 法拓扑优化理论	31
4.2.1 等效静态载荷法.....	31
4.2.2 BESO 拓扑优化方法	32
4.3 基于等效载荷法的 ABAQUS-MATLAB 动态拓扑优化集成方法	33
4.3.1 基于 BESO 法的 ABAQUS-MATLAB 拓扑优化集成方法	33
4.3.2 BESO 法在不同平台运行效率对比	34
4.3.3 基于等效静载荷法动态拓扑优化集成方法实现	35
4.4 两种吸能盒的刚度拓扑优化	36
4.4.1 冲击载荷下二维梁结构的拓扑优化.....	36
4.4.2 单材料吸能盒的碰撞模型及拓扑优化结果	37
4.4.3 Al/CFRP 混合结构吸能盒的优化结果及分析	38
4.4.4 不同碰撞角度下 Al/CFRP 混合结构吸能盒的优化结果...	41
4.5 本章小结.....	42
第五章 基于混合元胞自动机法耐撞性拓扑优化集成方法及应用 ...	45
5.1 引言.....	45
5.2 混合元胞自动机拓扑优化方法	45
5.2.1 混合元胞自动机的组成.....	46
5.2.2 全局和局部控制策略.....	47
5.2.3 更新规则.....	48
5.3 基于混合元胞自动机法的拓扑优化集成方法实现	48
5.3.1 初始化.....	48
5.3.2 有限元分析前处理.....	49
5.3.3 元胞自动机组建.....	50
5.3.4 有限元分析结果组装.....	51
5.3.5 挤压条件.....	51
5.3.6 单元状态更新.....	52
5.3.7 设计变量更新.....	52
5.4 吸能盒的耐撞性拓扑优化	52
5.4.1 不同更新规则对优化结果的影响.....	53
5.4.2 不同自动机邻域形状对优化结果的影响	55
5.4.3 吸能盒的耐撞性拓扑优化.....	57
5.5 本章小结.....	58

第六章 总结与展望	59
6.1 全文总结.....	59
6.2 研究展望.....	60
参考文献.....	61
附录 A.....	66
附录 B.....	70
附录 C.....	74
附录 D.....	78
附录 E.....	82

第一章 绪论

1.1 课题研究意义

结构优化是满足给定的几何约束、材料约束、状态约束和产品性能指标要求的重要设计方法之一，广泛用于航空航天、交通运输和装备制造等工程领域。根据结构几何形式的变化特点，优化问题可分为尺寸优化、形状优化和拓扑优化三类^[1]。相比于尺寸优化和形状优化，拓扑优化对结构初始设计构型的依赖性最小，在结构的概念设计阶段比前两者更具优势，而概念设计阶段是决定系统全寿命周期成本的最重要阶段。

轻量化设计是交通装备实现节能减排的重要途径之一。为实现碳达峰、碳中和“双碳”目标，国家把节能减排绿色发展作为重要的战略方向。国务院 2021 年发布的《“十四五”节能减排综合工作方案》提出了“大力推动节能减排，加快建立健全绿色低碳循环发展经济体系，助力实现碳达峰、碳中和”的目标，要求推广低能耗运输装备，提升新能源汽车在汽车销售总量中的占比。随着我国汽车保有量不断地增加，其对环境的影响也日渐突出，而轻量化作为降低结构重量的方法，是提高新能源汽车续航里程和降低燃油车排放的有效技术手段。同时，汽车碰撞安全性也是汽车企业和消费者重点关注的问题之一，提高碰撞安全性是汽车行业发展的必然选择。可见，节能与安全是当前汽车设计过程中必须要考虑的重要问题。然而，轻量化与碰撞安全性是一对相互矛盾的设计目标，但两者都可以通过优化实现，在汽车概念设计阶段采用拓扑优化能够使得轻量化和碰撞安全性达到平衡。因此，开展拓扑优化方法及其应用研究对提高汽车节能效果与碰撞安全至关重要。

当前拓扑优化方法的研究与应用主要围绕如下两方面开展：一是针对众多拓扑优化理论方法采用以 MATLAB 为代表的语言进行程序开发，此类程序通常是公开的基础代码，往往只能进行典型简单拓扑优化算例的分析，不能直接用于实际工况下的复杂系统拓扑优化设计，且相应功能拓展的难度大、周期长；二是针对实际工程要求使用以 Altair 公司的 OptiStruct 为代表的拓扑优化商业软件，进行复杂问题的静力、模态、屈曲和频响等分析，虽然其材料库和响应类型丰富，参数设置便捷，但是存在理论分析方法单一和不易二次开发的问题。可见，如何将现有众多拓扑优化新方法快速便捷的用于实际复杂系统的拓扑优化，对促进新方法的工程化应用具有重要意义。以 ABAQUS 为代表的商业有限元软件，能够进行接触碰撞、大变形和非线性等复杂问题的精确分析，且易于二次开发，操作界面友好。因此，开展拓扑优化基础程序与有限元软件的集成和应用研究，是推动拓扑优化方法发

展及其快速工程化应用亟需解决的问题。

本课题针对上述问题,以汽车吸能盒为研究对象,围绕拓扑优化方法与有限元软件的集成框架及其碰撞拓扑优化应用开展相关研究工作,对众多结构拓扑优化方法快速实现复杂问题的应用以及汽车节能与安全具有重要的现实意义和应用价值。

1.2 国内外研究现状及趋势

1.2.1 拓扑优化方法及基础程序

拓扑优化可以根据约束条件以及优化目标,通过对一个初始结构求解,使该结构演化成为一个无法通过预先假设形状的新结构。作为拓扑优化理论成型的标志,均匀化方法的提出距今不到四十年,得益于计算机软硬件的发展、材料科学的成熟以及众多研究人员的投入,拓扑优化得到了飞速的发展并取得了一系列的研究成果。在节能减排绿色发展的大背景下,高性能用料少的新型结构在各类工程问题中被广泛研究和应用。特别是随着增材制造等新型制造工艺被提出和应用,使新概念结构投入实际应用成为了可能。

在拓扑优化发展早期,许多学者为推动其发展做了很多研究工作,并取得了一些重要的研究成果。Lucien Schmit^[2]在 20 世纪 60 年代提出了将有限元分析和优化方法结合起来进行结构设计理念。Bendsøe 和 Kikuchi^[3]在 1988 年发表了使用均匀化方法在结构设计中生成最佳形状的开创性论文。此后, Hassani 和 Hinton^[4], Bendsøe 和 Sigmund^[5], Christensen 和 Klarbring^[6]等出版的具有较大影响力的书籍,对推动拓扑优化的发展起到的积极作用。均匀化方法给了许多学者启发,在此基础上分化出了多种拓扑优化方法,例如经典的变密度法^[7]、水平集方法^[8]、渐进结构优化法^[9]、相场法^[10]、拓扑导数法^[11]、冒泡法^[12]以及移动可变形组件法^[13]等。

随着拓扑优化理论的发展,许多用于拓扑优化的基础程序也相继被公开,这些基础程序实现了经典算例的拓扑优化。表 1.1 为部分开源程序的合集^[14]。由 Sigmund 编写的 99 行拓扑优化程序是最早公开的关于连续体拓扑优化的基础程序^[15]。随后其又公开了用于柔顺机构的 105 行代码和热传导问题的 91 行程序。Kharmanda 出于实现可靠性拓扑优化的目的,公开了一段 71 行的基础程序^[16]。Suresh 受 99 行代码的启发公开了一种跟踪帕累托最优曲线实现拓扑优化的 199 行基础程序^[17]。Liu 和 Tovar^[18]公开了一个 169 行的三维程序来解决三维拓扑优化问题,而 Schmidt 和 Schulz^[19]则公开了 2589 行的程序,使用 C 语言实现了三维线性弹性问题的拓扑优化。由于 99 行程序使用嵌套循环使得运算速度较慢,Andreassen 等人^[20]公开了比 99 行快两个数量级的 88 行基础程序。为了进一步提升运算效率, Ferrari^[21]提出了比 88 行更快的 top99neo 程序。

Huang 和 Xie^[22]公开了一种名为软杀伤的 BESO 基础程序,可用于解决简单的

二维柔度最小化拓扑优化问题。之后, Zhou 等人^[23]改进了该程序, 实现了具有理想运输性能的双相微结构复合材料拓扑优化。Xia 和 Da^[24]提出了一种用于具有光滑边界表示的扩展 BESO 方法, 命名为 ETO 方法并公开了实现该方法的基础程序。

当前拓扑优化水平集方法的基础程序, 可以分为两类: 一类是基于经典的水平集方法的基础程序, 例如 199 行^[25]、FreeFEM++^[26]、dLSM^[27]、FEniCS^[28]和 169 行^[29]; 另一类是基于参数化水平集方法的基础程序, 例如 88 行^[30]。此外, Zhang 等人^[31]提出了移动可变形组件法 (MMC), 并给出了 188 行的拓扑优化基础程序。还有使用非结构化多边形有限元网格进行拓扑优化的 PolyTop 程序^[32]、求解多材料拓扑优化问题的交替主动相位算法^[33]、不使用灵敏度的 PTO 算法^[34]等众多公开的基础程序。

表 1.1 部分公开发表的拓扑优化程序

年份	作者	语言	行数	拓扑优化方法
2001	Sigmund	MATLAB	99	固体各向同性材料惩罚法
2003	Bendsøe 和 Sigmund	MATLAB	105	固体各向同性材料惩罚法
2003	Bendsøe 和 Sigmund	MATLAB	91	固体各向同性材料惩罚法
2004	Wang 等人	MATLAB	199	水平集方法
2004	Kharmanda 等人	FEMLAB	71	固体各向同性材料惩罚法
2006	Olesen 等人	Scilab	111	基于密度的方法
2009	Allaire 等人	MATLAB	-	水平集方法
2010	Huang 和 Xie	MATLAB	101	双向渐进结构优化方法
2010	Suresh	MATLAB	199	固体各向同性材料惩罚法
2010	Challis	MATLAB	129	水平集方法
2011	Andreassen 等人	MATLAB	88	固体各向同性材料惩罚法
2011	Schmidt 和 Schulz	C/C++	2589	固体各向同性材料惩罚法
2012	Talischi 等人	MATLAB	190	固体各向同性材料惩罚法
2012	Zhou 等人	MATLAB	99	双向渐进结构优化方法
2014	Liu 和 Tovar	MATLAB	169	固体各向同性材料惩罚法
2014	Otomori	MATLAB	88	水平集方法
2014	Tavakoli 和 Mohseni	MATLAB	115	交替主动相位法
2015	Emre 和 To	MATLAB	90	比例拓扑优化法
2015	Xia 和 Breitkopf	MATLAB	119	均匀化方法
2015	Zegard 和 Paulino	MATLAB	-	基结构法
2015	Zuo 和 Xie	Python	100	双向渐进结构优化方法

表 1.1 (续)

年份	作者	语言	行数	拓扑优化方法
2016	Zhang 等人	MATLAB	188	移动变形组件法
2019	Dong 等人	MATLAB	149	均匀化方法
2019	Latorre	MATLAB	51	硬 0-1 进化优化方法
2019	Gao 等人	MATLAB	86	固体各向同性材料惩罚法
2020	Yaghmaei 等人	MATLAB	62	水平集方法
2021	Picelli 等人	MATLAB	101	二元结构拓扑优化方法
2021	Zhu 等人	FreeFEM	89	固体各向同性材料惩罚法
2021	Wang 和 Kang	MATLAB	80	水平集方法
2022	Zhuang 等人	MATLAB	172	双向渐进结构优化方法

新的拓扑优化方法不断涌现,相关方法的基础程序也相继被公开,例如,基于 SIMP 法的解决多尺度复合材料同时拓扑优化的 ConTop^[35]和用 FreeFEM 编写的 89 行^[36];实现了采用二元设计变量和形式化数学规划的 TOBS 法 101 行基础程序^[37];基于 BESO 法对三角形网格进行拓扑优化的 TriTOP172^[38],用于求解几何非线性拓扑优化的 137 行基础程序^[39];基于 LSM 法的应用密度滤波器求解的柔顺机构优化问题的 62 行基础程序^[40];采用速度设计变量和基函数构造速度场的 100 行基础程序^[41]等。

上述拓扑优化公开基础程序大多数是在 MATLAB 平台上运行,少数使用 C/C++语言、FreeFEM 和 FEniCS 求解器编写和运行,这些程序均可以在公开发表的论文中获得。通过分析若干公开的基础程序发现,除了与拓扑优化理论方法对应的基础程序外,在此基础上发展的程序主要是针对优化问题扩展、运算效率提升和不同优化对象等方面开展的算法理论研究。虽然学者对各种拓扑优化方法进行了拓展,但是因实际问题的复杂性,上述基础程序难以满足工程复杂系统的拓扑优化。

为了解决该问题,学者们开始尝试采用商业有限元软件与拓扑优化基础程序相结合的方式。ZUO 等人^[42]和任高晖^[43]对使用 ABAQUS 脚本接口实现双向渐进结构优化方法做了研究。Chen 等人^[44]采用 ANSYS 参数化设计语言提供高级有限元分析的方式,在 MATLAB 中基于 SIMP 法编写了一段 213 行的基础程序,实现了大位移弹性结构的拓扑优化。Papazafeiropoulos 等人^[45]公开了一种名为 A2M 的工具,将 ABAQUS 和 MATLAB 连接起来,可以实现结果后处理、统计分析和数学优化。借助 A2M 工具包,Antony^[46]和王朝辉^[47]进行了双向渐进结构优化方法的 ABAQUS-MATLAB 平台实现研究。然而,拓扑优化是一个循环迭代的过程,采用 ABAQUS 和 MATLAB 进行拓扑优化集成,二者之间需进行数据传递。A2M 工

具仅能进行 ABAQUS 到 MATLAB 的单向传递,且执行的对象为输入文本文件——.FIL 文件。分析发现,采用 A2M 进行 ABAQUS-MATLAB 拓扑优化集成存在如下两方面问题:(1).FIL 文件生成需要一定时间,且每一迭代步均需生成该文件,额外增加了拓扑优化过程的时间,大大降低了原拓扑优化方法的运行效率;(2) MATALB 向 ABAQUS 传递信息需要通过更改文本文件实现,会出现程序误读的现象,稳定性不高,增加了有限元模型的建模难度。

综上所述,自拓扑优化方法被提出以来,相继公开了大量的基础程序。这些公开基础程序往往只能进行简单经典拓扑优化问题的计算,虽然学者们对基础程序的功能拓展做了大量研究工作,由于实际问题的复杂性,上述基础程序仍难以直接用于工程复杂系统的拓扑优化设计,如碰撞拓扑优化问题。采用基础代码和商用软件集成拓扑的方式,为拓扑优化基础程序简便快捷的用于工程拓扑优化问题提供了可能,相关研究工作仍处在发展初期,还需进一步深入的研究。

1.2.2 碰撞拓扑优化方法

目前公开的拓扑优化基础程序往往是针对线性静力学拓扑优化问题进行的,为了实现动态非线性问题拓扑优化,学者们开展了一系列卓有成效的研究工作。Mayer 等人^[48]以内能最小化为目标,使用均匀化方法解决了某汽车结构的耐撞性问题。Forsberg 等人^[49]针对非线性的拓扑优化问题,将目标函数改为优化目标的内能密度。Pedersen^[50]在耐撞性拓扑优化设计中,提出了多个目标和约束条件,有效的控制了优化结构的能量耗散,从而减少了乘客受到的伤害。碰撞拓扑优化方法可分为四类,即:基结构方法(GSA)、图形和启发式准则技术(BGHT)、混合元胞自动机法(HCA)和等效静态载荷法(ESLO)。

基结构方法是一种为了避免复杂的碰撞行为,将设计空间以梁的宏观单元进行填充的一种优化方法。Pedersen^[50]使用了多种方法来删除或修改宏观单元以进行优化设计。Torstenfelt 和 Klarbring^[51]为了实现少量数值计算的情况下模拟非线性碰撞,基于材料属性提出了一个待优化的梁和节点组成的模型。Schumacher^[52]等人针对碰撞结构难以进行拓扑优化的问题,提出了在结构中依次引入孔洞,并根据优化需求改变孔洞位置和形状的方法。之后,这一想法被 Ortmann 和 Schumacher^[52]拓展到基于图像的方法中,采用启发式的规则来决定改变结构的形状,提出了图形和启发式准则技术。

Tovar^[53]等人使用混合元胞自动机的方法,通过引入内能目标值来控制设计域中单元的删减,从而获得了以耐撞性为目标的最优拓扑优化结构。Bandi 等人^[54]在耐撞性设计基础上,提出了根据应变能分区的方法,使最终的拓扑结构能量分布更加均匀。Masoud 等人^[55]通过引入可变邻域半径的概念,提高了 HCA 算法的搜索效率,使碰撞条件下所产生的能量被车辆结构更加均匀的吸收。

等效静态载荷法是一种兼顾动态和非线性特性的优化策略，它与早期多数碰撞拓扑优化研究中采用静态载荷来表示冲击响应有较大不同。等效载荷的思想最初是由 Choi 和 Park^[56]在处理线性动态响应的尺寸优化和形状优化时提出的，之后 Jang 等人^[57]提出了利用等效静载荷策略处理线性动态的拓扑优化问题。Lee 和 Park^[58]提出了一种改进的非线性动态拓扑优化方法，该方法适用于碰撞问题中的小幅度变形情况。针对大变形和大位移的拓扑优化问题，有的学者提出可以使用转换设计变量的方法来删除具有高等效应力的单元。等效静载荷方法在单次循环中要进行多次的有限元分析，因此该方法的时间成本较高。

碰撞拓扑优化方法的不断发展，使其逐渐被用于解决工程碰撞拓扑优化问题。Jang 和 Lee^[57]提出了一种使用正交阵列和等效静态载荷方法确定吸能盒界面尺寸的一种结构设计方案，并给出了三种新型的吸能盒。夏铭等人^[59]提出使用吸能盒的等效模型进行拓扑优化得到了体积约束下刚度最大化的优化结果。Lee 和 Park^[58]提出了一种基于等效静态载荷法的非线性动态拓扑优化方法，并使用该方法对吸能盒以耐撞性为目标进行了优化。Davoudi 和 Kim^[60]针对非线性塑性问题提出了一种薄壁吸能盒塑性铰链拓扑优化方法，得到了与原结构具有相同吸能能力的优化结果。Ren 等人^[61]为解决大变形碰撞条件下等效静态载荷法可能会失效的问题提出了一种考虑大变形和塑性屈曲的结构耐撞性拓扑优化方法，该方法为碰撞下吸能盒的结构设计提供了一种可行的策略。

综上所述，相比于静力学拓扑优化设计，碰撞拓扑优化因存在大位移、大变形等非线性问题，使其研究更具挑战性。目前碰撞拓扑优化方法的发展还不够成熟，仅有 ESLO、HCA、GSA 和 BGHT 等四种方法，它们各具特色各有应用，针对同一设计问题获得的碰撞拓扑构型也存在较大差异。上述研究往往是依托商业软件进行的，针对工程问题基于上述方法结合有限元软件开展碰撞拓扑优化研究工作，对促进碰撞拓扑优化方法的发展及其工程化应用，具有现实意义和实用价值。

1.3 ABAQUS 和 MATLAB 软件简介

ABAQUS 作为国际上最先进的大型通用非线性有限元分析软件之一，其具有强健的计算功能和广泛的模拟性能，拥有大量不同种类的单元模型、材料模型和分析过程等。无论是分析简单的线性弹性问题，还是包括几种不同材料、承受复杂的机械和热载荷过程，以及变化的解除条件的非线性组合问题；无论是分析静态和准静态问题，还是稳态和动态问题；无论是隐式求解，还是显示求解，应用 ABAQUS 计算分析都会得到令人满意的结果^[62]。

此外 ABAQUS 还为用户提供了几种二次开发手段，分别是：（1）通过修改环境初始化文件以改变 ABAQUS 的默认设置；（2）采用子程序接口开发新模型的方式，控制 ABAQUS 的计算过程；（3）利用 ABAQUS 图形用户接口工具包以创建

新的图形用户界面；(4) 运用 ABAQUS 脚本接口，进行有限元分析的前处理和后处理^[63]。

MATLAB 是一款由 The MathWorks 公司发行的商业数学软件。除了具有矩阵运算、图像绘制等数学软件的常规功能外，还可以用于创建用户界面和调用其他语言例如 C、C++、Java、Python 和 FORTRAN 等编写的程序。MATLAB 也具有自己的编程语言叫做 MATLAB 语言，是一种交互性的数学脚本语言，语法与 C 语言相似。支持逻辑、数值、文本、函数柄和异素数据容器在内的 15 中数据类型，并将每一种数据类型都定义为矩阵或者阵列的形式。

1.4 本文研究工作

本文围绕结构拓扑优化众多基础程序的快速工程应用问题，重点针对 ABAQUS-MATLAB 拓扑优化集成方法、通用 ABAQUS-MATLAB 拓扑优化集成方法、ABAQUS-MATLAB 碰撞拓扑优化集成方法实现及其工程应用等几方面开展研究工作，论文拟采用的总体技术路线，如图 1-1 所示。

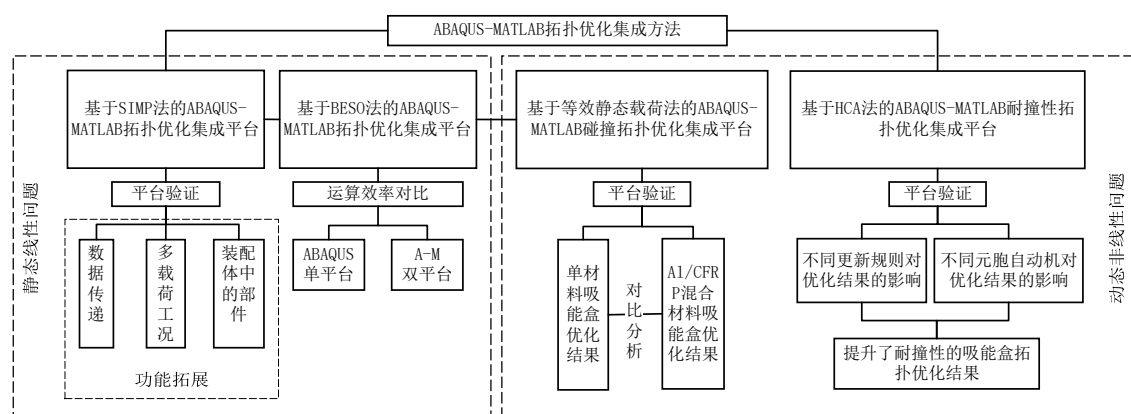


图 1-1 总体技术路线

首先，提出一种针对模型数据库文件的 ABAQUS-MATLAB 拓扑优化集成方法框架；在此基础上，进行基于 SIMP 法 ABAQUS-MATLAB 拓扑优化集成方法的实现和验证研究；之后，实现了基于 BESO 法和 ESLO 法的 ABAQUS-MATLAB 碰撞拓扑优化集成方法，开展汽车吸能盒刚度最大化碰撞拓扑优化研究；最后，结合 HCA 法，实现 ABAQUS-MATLAB 耐撞性拓扑优化集成方法，进行汽车吸能盒耐撞性拓扑优化研究。全文共有六个章节，各章节内容如下：

第一章，绪论。给出论文的研究背景和意义；回顾总结了结构拓扑优化方法及其基础程序的发展，剖析了存在的问题；综述了碰撞拓扑优化的发展及其应用情况；对课题涉及的 ABAQUS 和 MATLAB 软件做了简述；最后按照章节详细说明本论文的研究内容。

第二章，SIMP 法基本理论及 ABAQUS-MATLAB 集成方法框架研究。分析

SIMP 方法的特点及其与本集成框架方案间的关系；结合模型数据库文件以及拓扑优化 SIMP 法 99 行基础代码，提出了 ABAQUS-MATLAB 拓扑优化集成方法。

第三章，基于 SIMP 方法的 ABAQUS-MATLAB 拓扑优化集成方法实现所需核心程序的研究。根据上一章的集成框架，完成包括初始化、循环控制、设计变量更新、网格过滤的 MATLAB 核心程序和包括有限元分析初始化、设计变量组装、材料属性更新、后处理的 Python 核心程序，实现基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成方法；通过典型二维和三维算例计算，对拓扑优化集成方法进行验证；针对改善数据传递、多工况、装配体结构等问题对集成方法核心程序进行功能扩展。

第四章，基于 BESO 法和等效静载荷法的碰撞拓扑优化集成方法研究及汽车吸能盒刚度拓扑优化。采用本文提出的拓扑优化集成方法，实现基于 BESO 法的拓扑优化集成方法，比较 BESO 法在不同平台的运行效率；基于该集成方法引入等效静态载荷理论实现碰撞拓扑优化问题求解，对 AL/CFRP 吸能盒以刚度最大化为目标开展碰撞拓扑优化研究，并将优化结果与普通吸能盒的优化结果进行对比。

第五章，基于 HCA 法的 ABAQUS-MATLAB 耐撞性拓扑优化集成方法研究及汽车吸能盒的耐撞性拓扑优化。根据 HCA 法的基本理论，采用本文 ABAQUS-MATLAB 集成方法，实现了基于 HCA 法的耐撞性拓扑优化集成方法；分析了不同更新策略和自动机邻域对优化结果的影响，以耐撞性为目标对汽车吸能盒进行了拓扑优化。

第六章，总结与展望。归纳和总结了本文的研究工作，并对研究工作的不足之处进行了分析，对后续研究进行了展望。

第二章 ABAQUS-MATLAB 拓扑优化集成框架设计方案

2.1 引言

1988 年 Bendsoe 和 Kikuchi 提出了一种名为均匀化的拓扑优化方法,该方法将微结构引入到拓扑结构中,把微结构的几何参数当作设计变量,通过增加或者删除微结构的方式达到改变结构拓扑形状的目的。这一方法的提出极大地推动了拓扑优化方法研究的进展,变密度法、水平集法、相场法和演化方法相继被提出。其中 SIMP 法应用最为广泛,基于此方法 Sigmund 在 2001 年发布了 99 行 MATLAB 拓扑优化基础程序,该程序成为了后续很多拓扑优化基础程序开发的基础,对拓扑优化方法及程序研究产生了深远影响。

本章首先介绍 SIMP 法拓扑优化理论,以该方法为拓扑优化基本理论提出了 ABAQUS-MATLAB 拓扑优化集成框架设计方案。分析 ABAQUS 的文件类型和脚本接口确定集成的条件基础,基于经典拓扑优化 99 行 MATLAB 基础程序构建集成框架的 MATLAB 程序结构。

2.2 SIMP 法拓扑优化理论

为建立基于众多有限元法(FEM)类的拓扑优化方法集成方法,充分发挥拓扑优化基础代码和有限元软件的优势,开展 ABAQUS-MATLAB 拓扑优化集成方法研究。以该 ABAQUS-MATLAB 集成方法为基础,实现拓扑优化集成方法过程中需依托拓扑优化基本理论和基础代码。以 SIMP 法为代表的有限元类拓扑优化方法,具有以下几方面特点:

(1) 结合第 1 章所述拓扑优化理论发展历程, SIMP 方法自提出至今经过了长时间研究与应用证明了该方法成熟性;

(2) SIMP 方法与其他拓扑优化方法存在联系,许多优化方法是在 SIMP 法的流程基础上发展而来,实现了基于 SIMP 法的拓扑优化集成方法后,更容易将 ABAQUS-MATLAB 集成框架推广至其他拓扑优化方法;

(3) SIMP 法是一种结果含有灰度(即设计变量在 0-1 间分布)的拓扑优化方法,实现了设计变量连续的方法就更容易实现设计变量只有 0 和 1 的二元设计变量的拓扑优化方法。

基于上述原因,本文以 SIMP 方法为 ABAQUS-MATLAB 集成设计过程中的拓扑优化基本方法。

2.2.1 SIMP 法拓扑优化模型

为了将离散型问题转化为连续性优化问题, SIMP 法提出了一种引入中间密度

单元的方法进行计算。但在实际制造过程中，一个单元内只有填充材料和不填充材料两种状态，中间密度的存在使得模型无法制造。为此 SIMP 法还提出了一个通过对中间密度单元进行惩罚的方式来避免产生中间密度单元，使中间密度单元在迭代计算后趋向于实体单元或非实体单元。

基于 SIMP 法的材料插值模型为：

$$E(x_i) = E_{\min} + (x_i)^p (E_0 - E_{\min}), x_i \in [0, 1] \quad (2.1)$$

其中， $E(x_i)$ 是插值以后的弹性模量， E_0 是实体部分材料的弹性模量， E_{\min} 是为了防止刚度矩阵变得奇异而分配给孔洞区域的一个非常小的刚度。 x_i 是单元的相对密度，当 x_i 取 1 时表示有材料，取值为 0 时表示无材料填充也就是孔洞区域。 p 是惩罚因子。

在本文采用的材料差值模型表达式为：

$$E(x_i) = (x_i)^p E_0 \quad (2.2)$$

其中， x_i 为编号为 i 的单元的相对密度。

SIMP 法优化的数学模型为：

$$\begin{aligned} \text{find } X &= (x_1, x_2, x_3, \dots, x_i)^T \in R \\ &i = 1, 2, \dots, m \\ \min C(X) &= FU = U^T KU = \sum_{i=1}^m (x_i)^p u_i^T k_0 u_i \\ \text{s.t. } KU &= F \\ V &= fV_0 = \sum_{i=1}^m x_i v_i \\ 0 < x_{\min} &\leq x_i \leq x_{\max} \leq 1 \end{aligned} \quad (2.3)$$

该模型是求解在体积或者质量约束下的优化对象的最小柔度，最终得到在约束条件下的最大刚度的优化结果。其中， X 是单元相对密度矢量为单元设计变量， C 是结构柔度， F 和 U 分别是载荷矢量和位移矢量， k_i 为单元刚度矩阵， k_0 为初始单元刚度矩阵， v_i 为单元体积， f 是保留的体积分数， V_0 是初始体积， x_{\min} 是设计变量的下限值， x_{\max} 是设计单元的上限值， m 为最大单元数。

2.2.2 优化规则

SIMP 方法通过使用最优标准方法（OC method）来更新设计变量。设计变量的更新准则如下所示：

$$x_{\text{new}} = \begin{cases} \max(x_{\min}, x_e - m) \\ x_e B_e^\tau \\ \min(1, x_e + m) \end{cases} \quad (2.4)$$

此处的 τ 为数字阻尼系数，后续文中 τ 取值为 1/2。 m 是每次循环的最大移动上限。在

上述的数学表达式中：第一种情况是当 $x_e B_e^r$ 小于 x_{min} 或 $x_e - m$ 时，取 x_{min} 和 $x_e - m$ 之间的最大值；第二种情况是当 $x_e B_e^r$ 处于 x_{min} 、 l 和 $x_e - m$ 、 $x_e + m$ 之间时，选择更新变量 $x_e B_e^r$ ；最后一种情况是当 $x_e B_e^r$ 大于 l 或 $x_e + m$ 时，取 l 和 $x_e + m$ 之间的最小值。 B_e 通过最优标准方法计算得出：

$$B_e = \frac{-\frac{\partial c}{\partial x_e}}{\lambda \frac{\partial V}{\partial x_e}} \quad (2.5)$$

为满足体积约束， λ 作为拉格朗日算子其值等于二分法中的中间值。目标函数的灵敏度表达式通过目标函数 c 对设计变量 x_e 进行求导得到。

单元应变能 E_e 与式(2.7)等号右边各项的乘积的1/2等值，所以目标函数的灵敏度表达式也可以用含有 E_e 的算式表示：

$$E_e = \frac{1}{2} x_e^p u_e^T k_0 u_e \quad (2.6)$$

$$\frac{\partial c}{\partial x_e} = -U^T \frac{\partial K}{\partial x_e} U = -p(x_e)^{p-1} u_e^T k_0 u_e = -p \frac{2E_e}{x_e} \quad (2.7)$$

2.2.3 灵敏度过滤方案

在计算域内，材料密度为1的单元和材料密度为0的单元趋向规律性的分布状态成为棋盘格现象。此外，网格划分密度会对计算结果产生一定程度的影响，一般情况下网格划分密度越大越有可能出现很多细小的分支结构，这种现象叫做网格依赖性。棋盘格现象和网格依赖性都会降低拓扑优化结果的可制造性，使拓扑优化变得无意义。使用引入权重因子的方法对过滤半径以内的其他单元的灵敏度进行加权平均计算，最终能修正单元的灵敏度达到过滤的目的。依靠这种方法可以避免棋盘格现象并减少网格依赖性。目标函数对设计变量修正后的单元敏感度值为：

$$\frac{\partial \hat{c}}{\partial x_e} = \frac{\sum_{i=1}^N H_i x_i \frac{\partial c}{\partial x_i}}{x_e \sum_{i=1}^N H_i} \quad (2.8)$$

式中，卷积算子为：

$$H_i = r_{min} - dist(e, i), \{i \in N | dist(e, i) \leq r_{min}\} \quad (2.9)$$

此处 N 为总单元数目， r_{min} 为过滤半径， $dist(e, i)$ 为单元 e 和 i 中心坐标之间的距离。

2.3 ABAQUS-MATLAB 拓扑优化集成框架设计方案

由上一节 SIMP 方法理论可知，该拓扑优化方法为循环结构，在一次循环中包含有限元计算、有限元结果分析、灵敏度过滤以及设计变量更新等4个过程。根据

ABAQUS 可视化界面和有限元分析和 MATLAB 的矩阵计算能力的各自特点, 确定由 ABAQUS 执行有限元模型创建、有限元分析、有限元模型更新等工作, 而 MATLAB 完成主程序控制、灵敏度分析、网格过滤、设计变量更新等工作。与其他的 ABAQUS-MATLAB 集成方法相比, 本文的集成框架中参与集成的对象是.CAE 和.ODB 文件, 而不是.INP 和.FIL 文件, 采用本文集成方法即减少了有限元后处理花费的时间, 又充分利用了 ABAQUS 的可视化优势, 此外本文集成方法采用自编程序实现而非外部工具, 其灵活性较高, 不局限于一种优化方法, 具有更高的应用价值。本文提出的 ABAQUS-MATLAB 集成框架方案, 如图 2-1 所示。

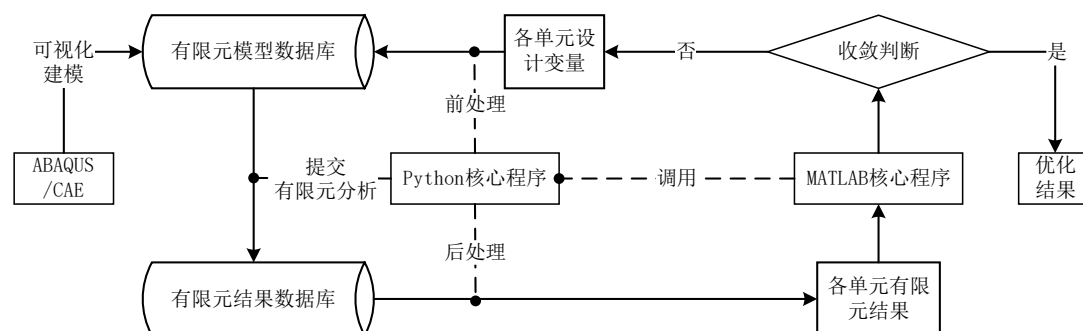


图 2-1 ABAQUS-MATLAB 集成框架设计方案

由图 2-1 可知, 在此方案中优化对象的有限元模型是基础, 基于 FEM 的 MATLAB 拓扑优化代码中, 对优化对象的有限元模型创建都是通过创建矩阵的形式实现的, 对于有限元模型的约束、载荷加载和有限元计算等都是通过向量及引入函数完成的, 因此, 拓扑优化基础代码及衍生代码只能处理形状规则、约束简单和载荷单一的对象。而 ABAQUS 可以创建甚至导入各种复杂形状的结构, 且具有友好的可视化界面, 设置约束和载荷方便。

在创建优化对象的有限元模型后即可开始运行优化程序, 优化程序的主流程由 MATLAB 核心程序控制, 在输入了优化参数后 MATLAB 核心程序开始执行初始化工作, 之后将优化对象的设计变量导出等待 Python 核心程序处理。Python 核心程序在读取设计变量后会执行有限元模型更新、提交求解、结果输出和有限元模型信息输出等工作。待 Python 核心程序执行完毕后, 得到所需要的有限元分析相关数据, 根据这些数据 MATLAB 核心程序即可继续执行 SIMP 法剩余的部分, 直到完成本次循环。

2.3.1 ABAQUS 文件类型与脚本接口

在 ABAQUS 中建立模型时需要处理的问题包括: 创建初始的有限元模型、根据设计域与非设计域的要求建立多个部件进行装配、根据预期单元数量划分合适的网格。建立有限元模型后, 其将保存到后缀为.CAE 格式的文件。虽然 ABAQUS 的.INP 后缀文件记录了全部的模型信息并且拓展性很强大, 但.INP 文件是在 job 命

令创建之后才生成的求解输入文件,出于可视化和简化更新模型操作的考虑,采用程序命令调用.CAE 文件的形式来实现更改模型,而非更改 INP 文件。完成 ABAQUS 中有限元前处理后,不仅会生成.CAE 文件和.INP 文件,还会生成后缀为.JNL 的模型日志文件,该文件记录了用户对有限元模型的操作信息。

在提交有限元求解过程中,ABAQUS 会生成后缀为.STA、.MSG 和.DAT 等类型的文件,分别为状态文件、信息文件和打印输出文件,这些文件对有限元求解过程进行了监测和记录。待有限元求解完成后分别生成.ODB 和.FIL 后缀文件,这两个文件中存储着计算结果信息,其中.ODB 文件为结果数据库文件,是 ABAQUS 在输出结果时默认输出的文件,在使用 ABAQUS/Viewer 浏览结果信息时所打开的即为该文件;而.FIL 文件则需在前处理输入关键字才会被输出的二进制文本文件,该文件的作用是便于其他程序读取。图 2-2 给出了随 ABAQUS 工作流程生成的文件类型。操作对象文件有.ODB (模型数据库文件)和.FIL 文件(文本文件),比较.ODB 和.FIL 文件可以发现:

- (1) ODB 文件与 FIL 文件中记录着同样的结果信息;
- (2) 通过 ABAQUS 脚本文件接口即可对 ODB 文件进行处理,而读取 FIL 文件则需要其他方法;
- (3) ODB 文件默认生成无需在前处理时添加操作,而 FIL 文件不仅需要添加关键字而且生成还需要时间。



图 2-2 随 ABAQUS 有限元计算流程中各文件生成顺序

整个集成方案为循环结构,因此,需要一种能够自动进行有限元前处理、求解计算、后处理的方法。ABAQUS 脚本接口(ASI)则正好可以解决这一问题。ASI 是以 Python 语言为基础所开发的,其功能有:对 ABAQUS 环境文件进行自定义、使用宏录制的方式批量进行前后处理、读取或者输出.ODB 文件、对有限元模型进行参数分析、创建 ABAQUS 子程序。脚本接口和 ABAQUS 内核的关系,如图 2-3 所示。通过可视化界面的命令行接口,GUI 工具包、脚本文件三形式都可以使脚本接口执行命令。根据脚本接口的特点,以脚本的形式调用 ABAQUS 脚本接口。由于 ABAQUS 脚本是基于 Python 开发的,因此,有限元过程的控制程序同样使用 Python 语言编写。

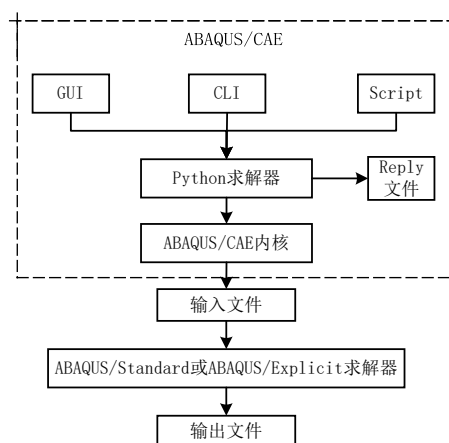


图 2-3 ABAQUS 脚本接口与 ABAQUS 内核的关系

2.3.2 集成框架中 MATLAB 程序结构

上一节以.CAE 和.ODB 文件为有限元对接文件，采用 Python 脚本控制有限元流程，解决了 ABAQUS 侧的问题。另一方面，还需要明确 MATLAB 程序中的流程结构。诸如 SIMP 法的众多有限元类拓扑优化方法大部分有公开的基础程序，因此，只需在原 MATLAB 基础程序上进行必要拓展即可满足集成需求。图 2-4 所示给出了 99 行 SIMP 法基础程序结构。

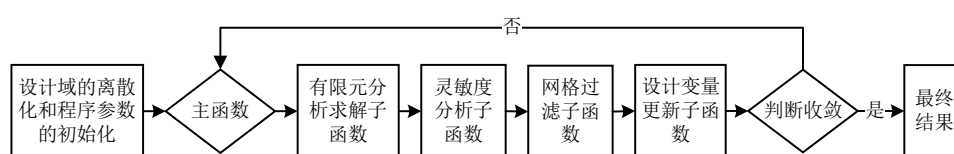


图 2-4 SIMP 法 99 行程序结构

99 行 SIMP 法基础程序为循环结构，整个流程以主函数调用各子函数的形式运行。解决 ABAQUS 与 MATLAB 两个平台间的数据交互问题，是建立 ABAQUS-MATLAB 拓扑优化集成框架的基础。在每一次的拓扑优化循环中 MATLAB 和 ABAQUS 之间需要各传递一次数据，数据主要是设计变量和有限元分析的结果，类型为数字结构。考虑到 Python 程序和 MATLAB 都有处理.xls 文件的能力，故采用 EXCEL 的.xls 文件作为两个软件之间的交互媒介，以解决数据交互问题。

分析 99 行 SIMP 法的基础程序结构可知，该代码是以矩阵类型的有限元模型为基础编写的，其有限元分析子程序和灵敏度分析子程序都是以矩阵计算的形式记录结果，因此需对上述两个部分做修改，以便对接 ABAQUS 有限模型。网格过滤子程序和有限元模型存在关联，该部分子函数也需作修改，而设计变量更新子函数则不需要做调整。用于集成的 MATLAB 核心程序结构，如图 2-5 所示，包含初始化设计、设计变量输出、Python 程序调用、有限元结果处理、灵敏度网格过滤、设计变量更新和收敛判断等 7 部分结构。

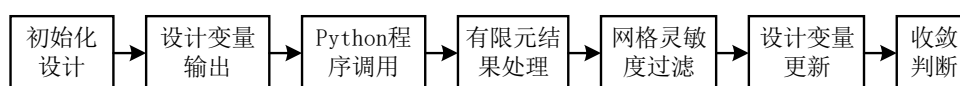


图 2-5 集成框架中 MATLAB 程序结构

2.4 本章小结

本章围绕 ABAQUS-MATLAB 集成框架的方案设计展开。首先对 SIMP 方法理论和优化模型进行了介绍,该方法具有发展成熟、应用基础广和使用连续伪密度的特点,有利于实现和验证 ABAQUS-MATLAB 拓扑优化集成方法。提出了使用 ABAQUS 创建模型、Python 程序控制有限元前后处理、MATLAB 程序控制拓扑优化流程的 ABAQUS-MATLAB 拓扑优化集成框架。分析了 ABAQUS 软件自带的脚本接口具有二次开发的优势,.CAE 和.ODB 文件是生成.INP 文件和.FIL 文件的基础,相比于文本文件,采用前者(模型数据库文件)进行集成,其稳定性更好、运行效率更高。针对经典 99 行 SIMP 法基础程序,设计了集成框架中 MATLAB 程序的结构,为 ABAQUS-MATLAB 拓扑优化集成方法的实现奠定了基础。

第三章 基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成方法

3.1 引言

采用 A2M 工具包可进行 ABAQUS 有限元结果的统计分析、后处理和数学优化,实现 ABAQUS 与 MATLAB 之间的数据传递,基于 A2M 工具包相关 ABAQUS-MATLAB 集成拓扑优化研究,主要是.INP 和.FIL 等文本文件为操作对象,如上一节所述,这些文本文件会降低集成方法核心程序的稳定性、增加其运行耗时。另一方面,采用 ABAQUS 脚本接口实现的基于 ABAQUS 平台的拓扑优化方法,同样存在运行效率问题,如采用 Python 语言编写的基于 BESO 方法的 ABAQUS 平台拓扑优化。因此,有必要建立运行效率高、稳定性好的拓扑优化集成方法核心程序,实现拓扑优化基础代码的快速工程应用。

本章围绕基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成方法的实现问题,详细阐述该集成方法的 99 行 MATLAB 程序和 88 行 Python 程序两个核心程序;对拓扑优化集成方法的有效性进行验证,针对数据传递、多载荷工况等问题对集成方法核心程序进行功能扩展。

3.2 MATLAB 核心程序及解读

集成方法的 MATLAB 核心程序,如附录 A 中表 A-1 所示,该程序全长共有 99 行,是由经典的 SIMP 法 99 行基础程序改进而来。该程序采用主函数调用子函数的形式,主要包括初始化阶段、循环控制、网格过滤和设计变量更新等功能。本节按照运行顺序对该核心程序做详细解释。

3.2.1 初始化部分

在初始化部分,前两行代码为集成方法的备注,运行集成方法需要将 MATLAB 核心程序、Python 核心程序和 ABAQUS 有限元模型放在同一文件夹下,将该文件夹称为工作目录。3-18 行代码为初始化主要内容,首先定义了主函数,主函数由函数名和函数参数两部分组成。主函数的名称为 topMA,需要注意的是此处的函数名需和文件名一致。主函数所需参数和优化参数相关,依次为优化模型的最大单元数、最终体积分、惩罚因数和过滤半径(第 3 行)。

为了 ABAQUS 脚本接口能够读取到有限元模型,需要在 MATLAB 代码初始化阶段输入模型信息。模型信息以变量的形式记录在程序中(第 4-6 行),三个变量代表的信息分别为模型数据文件所在路径、模型数据库文件名称和模型名称。每次更改模型时只需将对应参数内容进行更改,需要说明的是输入 CAE 文件名时不需要添加后缀。采用文本文件(.txt)的形式输出模型信息,该文本文件能以任何名称命名,但是需要与 Python 程序中的名称保持一致,在示例程序中对模型信息

文本命名为 `modelData`。

使用 `strvcat` 函数将 3 个和模型信息有关的变量组成列向量，之后将这其保存到模型信息文本中。每次循环迭代都需要进行结果显示，但由于拓扑优化的迭代次数较多且 MATLAB 的显示区域有限，因此需要将每步迭代的信息保存。迭代信息的记录载体仍然选择文本文件，因此在初始化时就创建了一个名为 `result-enum` 的文本文件等待被写入（第 7-14 行）。

因有限元模型由 ABAQUS 创建，而设计变量既要参与有限元模型的材料属性更新又要参与拓扑优化的计算，故在初始化阶段需要创建初始设计变量。创建一个行数为有限元模型网格单元总数，列为 1 的单位向量，根据 SIMP 理论该向量中的所有数值需要乘以目标体积分数，将该向量命名为 x 。创建名为 `EmlN` 的向量，以确保在数据传递时每个有限元单元的设计变量与设计变量相匹配。该向量内的数值从 1 开始直到有限元模型网格的单元最大数量为止。向量中的每一个数字代表的都是一个有限元单元，数值即为有限元网格编号。创建变量 `loop`，该变量存储的是拓扑优化当前的迭代次数，在初始化部分该变量从 0 开始。创建变量 `change`，该变量存储的是循环结束所需要满足的差值大小，将该变量赋值为 1（第 14-18 行）。

3.2.2 循环控制部分

在完成初始化的工作后，程序进入拓扑优化循环部分。使用 `while` 循环在差值 `change` 大于某一数值时，循环将一直运行，直到 `change` 小于这一数值。考虑到拓扑优化结果的精度，示例程序中的数值设定为 0.01（第 20 行）。在循环之初对变量 `loop` 进行更新（第 21 行）。循环次数对于 ABAQUS 中的有限元模型更新和 Python 程序是一个重要的信息，因此，需要将变量 `loop` 的值输出。使用 `fopen` 函数创建一名为 `loop.txt` 的文本文件，再利用 `fprintf` 函数将变量 `loop` 写入该文件中，最后采用 `fclose` 函数保存文本文件（第 22-24 行）。创建名为 `xold` 的变量，将上一次循环中的设计变量 x 保存在该变量中以便于在循环结束时计算本次设计变量的差值，在第一次循环时 `xold` 储存初始化时创建的变量 x （第 25 行）。

进入工作目录，该工作目录的路径即为初始化时输入的路径信息（第 26 行）。使用 `xlswrite` 函数进行设计变量输出，`xlswrite` 是 MATLAB 自带的函数，功能是向 EXCEL 电子表格中写入信息。输出该电子表格的目的是等待 Python 程序读取，因此，两个核心程序中存储设计变量的 EXCEL 电子表格命名要一致。当一个工作目录中有多个有限元模型时，为避免程序将优化对象混淆，故 EXCEL 电子表格命名为 `ExM_Evo` 加上最大单元数。如果工作目录中没有这一名称的电子表格，`xlswrite` 则会创建电子表格并写入，如果存在同一名称的电子表格，该函数将输出内容覆盖表格中原有内容。选择 `EmlN` 即单元编号为输出对象。根据 `loop` 迭代次数的不同，将内容输出到不同的工作簿中，以保留每次迭代中设计变量的信息，便于查看及对

比。

在工作簿中 *EmlN* 起始位置为第一行的第一列，结束位置取决于单元数量。与该向量相对应，设计变量 x 也以同样的方式输出到电子表格。位置为每一次循环工作簿的第二行第一列开始。至此单元编号与设计变量输出完毕，在 EXCEL 电子表格工作簿中每一行都为单元编号与该单元对应的设计变量。由于使用 `xlswrite` 函数时会导致命令行窗口有多余的信息出现，进而影响记录迭代信息，因此，使用特定语句将该提示关闭（第 27-29 行）。

采用 `system` 函数调用系统命令提示符，执行括号内的语句。该语句的作用是调用系统命令使 ABAQUS 内核执行指定的.py 文件，效果与在 ABAQUS 用户界面选择执行脚本再选择.py 文件相同（第 30 行）。

在 ABAQUS 执行 Python 程序时，第 30 行语句处于未完成的状态，因此 MATLAB 中的程序会停止运行直到第 30 语句执行完毕。待 ABAQUS 完成有限元分析任务后，MATLAB 核心程序继续执行。创建名为 c 的变量，该变量即为拓扑优化的优化目标，该优化目标的数值来自 ABAQUS 有限元分析结果，未读取有限元结果时 c 的值为 0。创建变量 dc ，储存每个单元的灵敏度，与优化目标相同，单元的灵敏度也由有限元分析结果得到，因此初始的 dc 向量为零向量（第 31-32 行）。

利用 `xlsread` 函数读取有限元分析的结果，**结果内容包括单元编号、单元编号对应的单元灵敏度和优化目标**。上述有限元分析结果以矩阵的形式赋予变量 *ELSE*（第 33 行）。优化目标值存储在矩阵 *ELSE* 第三行第一列。遍历矩阵 *ELSE* 的第二列数据即单元的灵敏度，根据式（2.7）组装每个单元的灵敏度，组装后的灵敏度根据单元编号存储到向量 dc 中（第 35-37 行）。

根据 SIMP 法拓扑优化流程，需要对灵敏度进行过滤以避免网格依赖，再由过滤后的灵敏度进行设计变量更新。在主函数中这两部分的工作均通过调用子函数实现，两个子函数在主函数后进行创建（详见 3.2.3 和 3.2.4 节）。调用网格过滤子程序进行网格过滤，调用设计变量更新子程序更新设计变量（第 38-39 行）。利用新旧设计变量计算差值，作为判断循环是否收敛的准则（第 40 行）。输出每次迭代运行的结果，该程序中每次迭代优化结果显示共四部分：分别为当前迭代次数、目标函数值、当前体积分数以及当前差值（第 41-43 行）。为了在程序关闭后仍能查看拓扑优化结果，因此在每次迭代时也将上述拓扑优化信息输入到名为 `result-elenum` 的文本文件中（第 45-48 行）。至此 MATLAB 核心程序主函数结束。

3.2.3 设计变量更新子函数

在完成网格灵敏度过滤后，主函数调用 OC 优化子函数更新设计变量。子函数的位置并不影响调用，考虑到设计变量更新子函数篇幅较小因此将其放在网格过滤子函数之前。本节将按照代码的顺序先介绍设计变量更新子函数。在子函数内容

之前,有一行该子函数名称的注释便于定位到该子函数(第 49 行)。子函数的名称为 OC,需要的参数依次为:有限元模型最大单元数、设计变量、目标体积分数和单元灵敏度(第 50 行)。变量 $l1$ 和 $l2$ 分别是设计变量更新的左边界和右边界,取值分别是 0 和 100000,变量 $move$ 是更新幅度,取值为 0.2(第 51 行)。当左右边界差值大于 0.0001 时则执行 while 循环程序, $lmid$ 是左右边界的中间值,根据式(2.4)对向量 x 中的所有设计变量进行更新,新的设计变量存储到向量 $xnew$ 中(第 52-54 行)。目标体积分数和最大单元数的乘积即为目标体积,当前设计变量的数值相加即为当前的体积,如果当前的体积大于目标体积则左边界 $l1$ 更新为中间值 $lmid$,反之,右边界 $l2$ 将更新为中间值 mid 。循环直到左右边界的差值达到目标,即达到体积约束则停止循环,结束执行该子函数。

3.2.4 网格过滤子函数

程序 62-99 行为网格过滤子程序。子函数的函数名为 dcn ,其输入参数有 $maxElNum$ 、 $rmin$ 、 x 和 dc ,分别对应设计域内的最大单元数、过滤半径、设计变量向量和单元灵敏度向量(第 63 行)。SIMP 法在 ABAQUS-MATLAB 平台运行与原 99 行基础程序在 MATLAB 平台运行存在差异,原 99 行基础程序中各单元以矩阵的形式排列,因此矩阵的行列号即可当作每个单元的中心坐标。而在 ABAQUS-MATLAB 集成框架中,有限元模型存在于 ABAQUS 中,因此 Python 程序在输出有限元结果时,需输出设计域的单元坐标。

采用 $xlsread$ 函数读取名为 $EL_coordinateS$ 的 EXCEL 电子表格文件,将读取的矩阵命名为 $oData$ 。矩阵 $oData$ 的列数共有四列,分别对应单元编号、单元 X 坐标、单元 Y 坐标和单元 Z 坐标,矩阵的最大行数由有限元模型的单元数决定。以矩阵 $oData$ 为对象使用 max 和 min 函数,这两个函数的返回值为矩阵每一列中的最大值和最小值,将返回值分别赋予变量 $vMax$ 、 $vMin$,两个变量都为 1 行 4 列的向量(第 65-66 行)。如果向量 $vMax$ 和 $vMin$ 中的第四个值相等,即有限元模型中所有单元的 Z 坐标都相等,说明该模型为二维模型,不然则为三维模型。如果是三维模型则对变量 p 赋值 4,反之如果是二维模型则对变量 p 赋值 3(第 67-71 行)。

$numArrays$ 的值代表参与网格过滤计算的维度。创建元胞数组 S ,该数组的行数与 $numArrays$ 有关。元胞数组为 MATLAB 中的一种数据类型,采用该数据类型存储坐标数据和筛选符合条件的单元。创建名为 dcn 的零向量用于储存过滤后的单元灵敏度(第 72-74 行)。

按照单元编号从第一个单元开始进行遍历,针对每一个单元进行以下重复的操作。创建变量 $suma$ 并赋值为 0,该变量代表着灵敏度因子,在遍历循环之初数值为 0,循环完毕后进行设计变量更新(第 75-76 行)。变量 $major$ 取矩阵 $oData$ 中的第 i 行, i 为单元的编号,即该变量为当前单元的坐标信息(第 77 行)。从 2 到

p 值间遍历单元的 X、Y 和 Z 坐标。分别在各坐标轴使用 find 函数查找过滤半径内的单元，find 函数的功能是返回符合查找条件的索引值。以查找 X 轴坐标为例，major 变量中的单元为主单元，在过滤半径内的单元为次级单元，查找范围的上限为主单元 X 轴坐标加上过滤半径，查找范围下限为主单元 x 轴坐标减去过滤半径。以图 3-1 中第 427 号单元为例，在该次循环中主单元为 427 号单元，再以 X 轴坐标查找时，次级单元为区域 1 中的所有单元。Y 轴和 Z 轴同样按照 X 轴的方式进行查找，每一个坐标轴的查找结果被保存在元胞数组 S 中（第 78-81 行）。

当有限元模型是二维模型时则数组 S 中共有两个元胞，分别是主单元 X 轴查找半径内单元编号和主单元 Y 轴查找半径内单元编号，对应图 3-1 中的区域 1 和区域 2。同时，对变量 S 中的元胞采用 intersect 函数，该函数的作用是取两个数组的交集，如图 3-1 所示，取区域 1 和区域 2 相交的区域 3，该区域即为主单元的过滤范围。如果是三维图形则是主单元 X 轴查找范围内单元编号，先与 Y 轴范围内单元对比编号取交集，再将交集结果与 Z 轴查找范围内的单元编号取交集，得到最终的过滤半径内所有单元的编号。

过滤半径内的所有单元编号保存在变量 *eleO* 中（第 82-86 行）。遍历 *eleO* 中的各元素，即遍历主单元过滤半径内的所有单元，将每个遍历到的单元称为次级单元，使用 sqrt 函数进行开平方根函数计算，得到主单元和次级单元之间的距离（第 87-89 行）。再以过滤半径为筛选条件，如果计算得到的距离小于过滤半径，则将该值与 suma 相加更新为式（2.9）中的因子。在遍历完过滤半径内所有的单元后根据式（2.8）得到主单元过滤后的灵敏度（第 89-99 行）。至此完成了当前主单元的灵敏度过滤，程序将按照单元序列以下一个单元为主单元进行过滤。

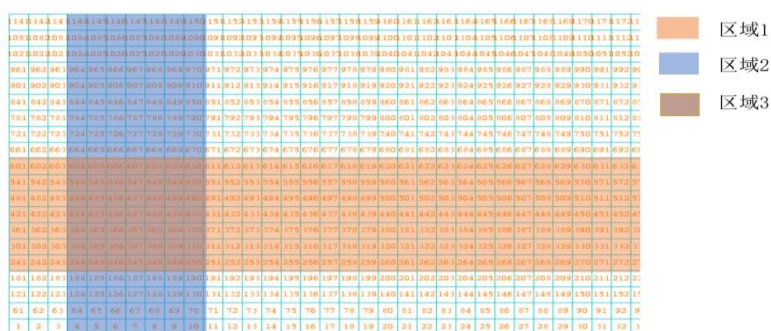


图 3-1 网络过滤范围示意图

3.3 Python 核心程序及解读

Python 核心程序由 MATLAB 核心程序第 30 行调用，该程序的作用是调用 ABAQUS 内核执行有限元分析相关的工作。附录 B 的表 B-1 给出了集成方法的 Python 核心代码，其运行流程如图 3-2 所示，主要包括三部分内容，即：针对.CAE 对象的有限元前处理、有限元求解计算、针对.ODB 对象的有限元后处理。Python

核心程序总长 88 行，后文将按照初始化、设计变量组装子函数、有限元模型载入、材料属性更新、有限元分析后处理等五部分进行阐述。

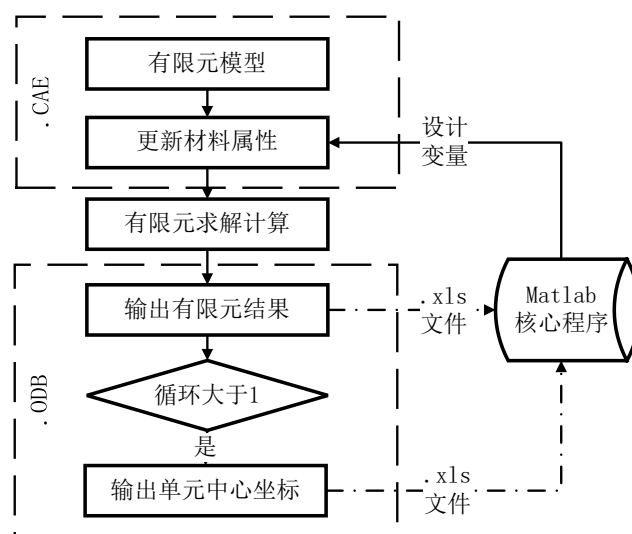


图 3-2 Python 核心代码流程图

3.3.1 有限元分析初始化

和众多的 Python 程序一样，该 88 行 Python 核心代码也在开头位置导入模块。所导入的模块包括查看 .ODB 文件的 openOdb 模块、读取 .xls 文件的 xlrd 模块、写入 .xls 文件的 xlwt 模块、系统命令 os 模块和分析步处理 step 模块（第 1-5 行）。其中，xlrd 和 xlwt 模块并不能直接导入，ABAQUS 脚本接口采用 Python2.7 版本，该版本无 xlrd 和 xlwt 模块包，因此，需要下载和安装上述模块包，以便进行调用。安装步骤为下载对应 Python 版本的模块包，然后将模块包放到 Python 根目录下，调用系统命令利用 pip install 进行安装。

根据系统命令函数 os.getcwd 获得 Python 程序所在的拓扑优化工作目录位置。选择该目录中的 modelData 文本文档传递给 Mas，由 readlines 将向量 Mas 中的数据传递给变量 data。利用 strip 方法将 data 中的内容分为路径名称、CAE 文件名称和模型文件名称等三部分，再将这些信息传递给变量 path、caeName 和 modelName（第 6-9 行）。读取名为 loop 的文本文档并传递给变量 loop，该变量用于记录迭代次数，由于 Python 语法的计数规则是从零开始，需减去 1 以对应正确的循环文件（第 10-12 行）。

3.3.2 设计变量组装子函数

与 MATLAB 语言不同，Python 语言需要先创建子函数才能调用。虽然在一次循环中每个设计变量只需要与单元编号对应一次，为使 Python 代码整体结构清晰，将设计变量组装部分定义为一个子函数。该子函数的名称为 get_data，函数有 dirCase 和 sheetNum 两个参数，分别代表着设计变量所在 EXCEL 表格文件的名称

和相应工作簿的编号（第 13-14 行）。采用 `xlrd` 模块中的 `open_workbook` 方法打开目标表格文件，将该文件传递给变量 `data`；根据工作簿代码定位到目标工作簿并传递给 `table` 变量，分别创建 `nor` 和 `nol` 两个变量，分别对应 `table` 中的行号和列号（第 15-18 行）。创建名为 `dict0` 的空字典结构，字典结构为 Python 语法中一种常见的数据类型，特点是其中的数据以键值对的形式成对保存（第 19 行）。以 `table` 变量中每一行之间的遍历作为主循环，每一列之间的遍历作为次循环（第 20-21 行）。在每一行中第一列的数字传递到变量 `eNum`，第二列数字传递到变量 `xDen`，上述两个变量分别表示单元编号和单元设计变量（第 22-23 行）。将每一行的 `eNum` 变量和 `xDen` 变量存储到字典 `dict0` 中，其中前者为键（key），后者是值（value），子函数运行完之后返回字典 `dict0`（24-25 行）。

3.3.3 有限元模型载入

以 `if_name=='_main_'` 作为主函数的开头（第 26-27 行）。ABAQUS 模型数据库中各对象以层级的形式存在，各对象之间的关系如图 3-3 所示。采用 `openMdb` 方法打开变量 `caeName` 所指代的模型数据库文件，并将该文件传递给变量 `myMdb`。运用 `models` 方法在模型数据中定位到变量 `modelName` 所指代的模型并传递给变量 `mdl`。同样，利用 `parts` 方法定位到目标部件并传递到变量 `part`，因为拓扑优化的对象为单个部件，部件名称不需要在外部导入或者输入，使用创建第一个部件后的默认名称 ‘Part-1’ 即可。故对装配体中某一部件进行拓扑优化时需要先创建该部件，或者将该部件命名为 ‘Part-1’（第 28-30 行）。

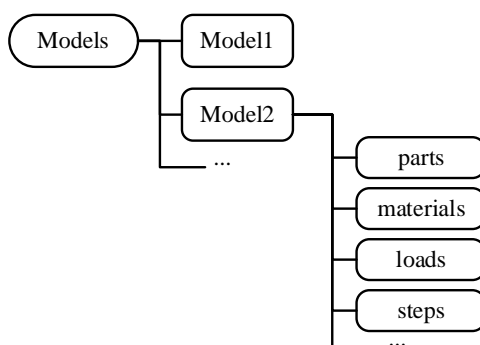


图 3-3 ABAQUS 模型数据库结构示意图

对 `part` 对象采用 `elements` 方法和 `nodes` 方法，以返回部件中的单元信息和节点信息分别至 `ems` 变量和 `nds` 变量（第 31 行）。创建空字典 `dict1` 和 `dict2`，调用 `get_data` 子函数并遍历其中的结果，将遍历到的键值对更新到字典 `dict1` 中（第 32-35 行）。遍历字典 `dict1` 中的所有键和值，在 `dict1` 中将每个单元编号作为键，各单元编号对应的设计变量作为值，如图 3-4 中第一行所示。在优化过程中每个单元的单元编号都是唯一的，而设计变量则有可能重复，另一方面，单元设计变量可能是

从 0.001 到 1 之间的任意数字, 因此缩短原有的键值对结构, 有利于材料属性更新和加速程序运行。遍历 *dict1* 中的键和值, 如果存在重复的值, 则生成一新的键值对将值作为键, 所有重复的单元编号组成列表作为新键值对的值, 没有重复的键值对则只交换键和值的位置, 如图 3-4 中第二行所示, 所有新的键值对都存储到字典 *dict2* 中 (第 36-40 行)。

```
{ '1' :0.2, '2' :1.0, '3' :0.2, '4' :1.0, '5' :0.7, '6' :1.0}
{ '0.2' :[1, 3], '1.0' :[2, 4, 6], '0.7' :[5]}
```

图 3-4 设计变量字典结构转换前后示意图

3.3.4 材料属性更新

变量 *penal* 指代惩罚因子, 初始设定值为 3, 根据优化需要进行更改。变量 *youngOld* 代表杨氏模量的初始值, 在该示例中使用的初始值为 1 (第 41-42 行)。创建 *indexMaterial* 变量用以材料属性命名, 其初始值从 0 开始。将 *dict2* 中的所有键提取并转化为列表传递给变量 *listDx* (第 43-44 行)。根据字典 *dict2* 中的元素数量进行遍历, 取当前设计变量的值传递给 *j*, 利用式 (2.2) 计算新的杨氏模量, 即变量 *young*。更新变量 *indexMaterial* 并从数字型转化为字符型, 同样的将变量 *j* 所对应的单元编号列表, 从数字型转化为字符型 (第 45-50 行)。

通过 *Material* 方法创建材料属性, 命名规则为 'Material' 加材料编号 *l*, 运用 *Elastic* 方法将更新后的杨氏模量赋给 *young*, 泊松比赋值为 0.3。采取 *Homogeneous* 方法创建截面属性, 命名规则与材料属性相同, 选择同一编号的材料属性作为对象 (第 51-52 行)。使用 *SectionAssignment* 方法赋予材料属性, 对象为同编号的截面属性, 材料赋予目标为列表 *ut* 中的所有单元 (第 53 行)。利用 *FieldOutput* 方法创建场变量输出, 名称为 'SEDensity', 分析步为 'Step-1', 输出关键词为 'ELSE', 即式 (2.7) 中的弹性应变能。采用 *HistoryOutput* 方法创建历史变量输出, 名称为 'ExtWork', 分析步名称是 'Step-1', 输出关键词为 ALLWK, 即式 (2.3) 中目标函数的数值 (第 54-55 行)。

利用 *Job* 方法创建分析作业, 为了区分和保留每次循环中有限元分析结果, 在创建分析作业时使用循环数作为名称后缀。而后调用 *submit* 方法将创建好的分析作业提交运算, 采用 *waitForCompletion* 方法等待作业分析结束 (第 57-59 行)。

3.3.5 有限元后处理

ABAQUS 的结果数据库文件与模型数据库文件相似, 其中的对象以层级结构存在, 结果数据库文件结构如图 3-5 所示。采用 *openOdb* 方法打开结果数据库文件, 结果数据库文件的命名与作业名一致仅后缀不同 (第 59 行)。根据图 3-5 中的层级结构, 先后调用 *steps* 方法、*frames* 方法和 *fieldOutputs* 方法, 从分析步进入最后一帧得到单元应变能, 再通过 *values* 方法得到单元应变能的具体数值传递给变

量 *elseV*。同样的在 *steps* 方法下运用 *historyRegions* 和 *historyOutputs* 方法得到 ALLWK 所代表的外力功，使用 *data* 方法得到具体数值并传递给变量 *obj*（第 60-61 行）。采用 *xlwt* 模块下的 *workbook* 方法和 *add_sheet* 方法创建工作簿（第 62-63 行）。创建空字典 *rawDc* 后遍历变量 *elseV* 中所有元素，将单元编号作为键、单元的弹性应变能作为值，二者合并成键值对存储到 *rawDc* 中。将 *rawDc* 中的所有的键和值分别生成列表 *num* 和 *val*，根据单元编号顺序将单元编号写入工作簿的第一列、单元的弹性应变能写入第二列，变量 *obj* 只有一个数值写入第一行第三列（第 64-74 行）。

如果当前为第一次迭代，即变量 *loop* 的值为 0 时，则输出单元中心坐标。在输出坐标之前采用与前文相同的方式创建工作簿（第 76-78 行）。创建空字典 *c0* 后，从第一个单元开始遍历，先利用 *connectivity* 方法得到单元所有的节点，再采用 *coordinates* 方法得到所有单元的坐标，通过计算最终得到单元的中心坐标（第 79-82 行）。将单元编号写入工作簿的第 1 列，X 轴、Y 轴和 Z 轴坐标依次写入第 2 到 4 列后保存 EXCEL 表格文件（第 83-88 行）。

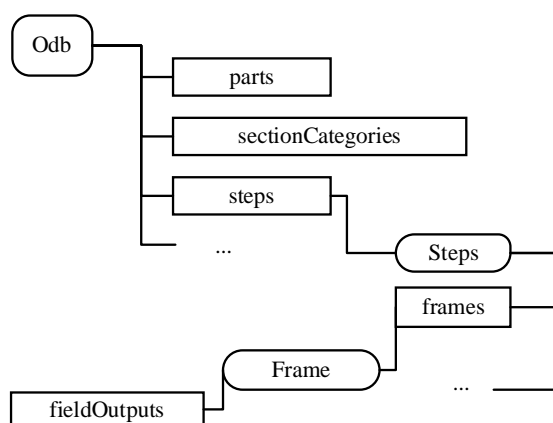


图 3-5 ABAQUS 结果数据库文件结构

3.4 拓扑优化算例及功能扩展

3.4.1 二维和三维算例

创建一个文件夹作为工作目录，将上述 MATLAB 核心程序文件和 Python 核心程序文件置于该工作目录中。根据图 3-6（a）的模型示意图，创建一个长度为 60mm、宽为 20mm 的二维梁结构。该模型为固支梁的一半，因此在施加约束时，梁的左侧施加对称约束而梁的右下角施加完全固定约束。在半梁的右上角施加向下的载荷。设定每个网格边长为 1mm、类型为四节点四边形网格，共有 1200 个网格单元，如图 3-6（b）所示。完成上述步骤后，将该模型数据库文件保存并退出 ABAQUS。将该模型的名称、工作目录、模型数据库名称、网格单元数等信息输入到 MATLAB 核心程序中，提交拓扑优化，经过 68 次迭代后最终的结果如图 3-6

(c) 所示, 其构型与图 3-6 (d) 原始 MATLAB 基础程序相应的拓扑优化结果一致, 表明了拓扑优化集成方法的正确性。

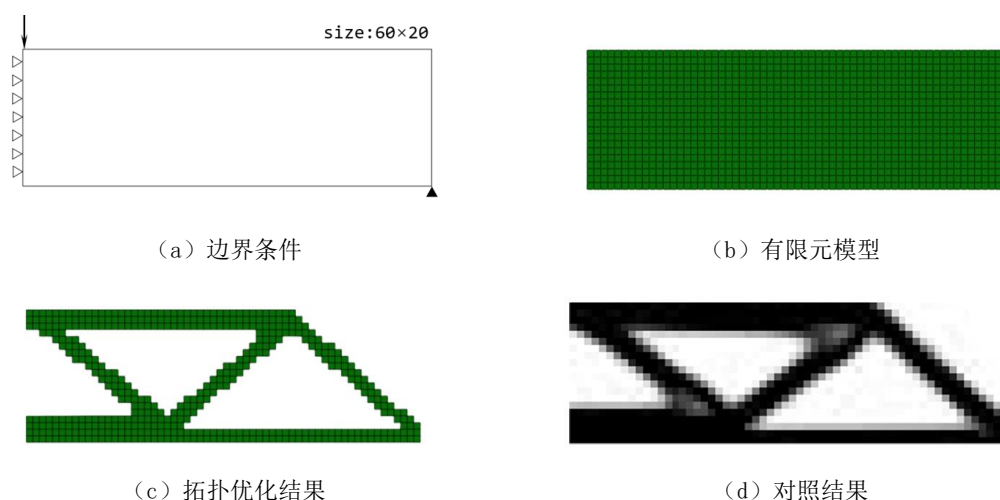


图 3-6 二维固支梁优化前后示意图

利用本文提出的拓扑优化集成方法对三维 L 型梁进行拓扑优化, 以进一步验证集成方法的有效性。如图 3-7 (a) 所示, 该梁外侧 L 型边长为 60mm, 内侧 L 型边长为 40mm, 梁的顶部与右侧为边长为 20mm 的正方形。在梁的顶部施加完全固定约束, 右侧施加向下的载荷。采用边长为 1 的八节点正方体网格进行网格剖分, 单元数量为 40000 个。设定目标体积分数为 0.2、惩罚因数为 3、过滤半径为 2.0。经过 59 次迭代后收敛得到优化结果, 如图 3-7 (b) 所示。在结果中 L 型梁的上部分多数材料被删除, 在转角处保留了 L 型梁两侧的材料, 接近载荷施加部位材料越向中间集中, 该结果与图 3-7 (c) 原 SIMP 法的三维程序相应结果构型吻合。

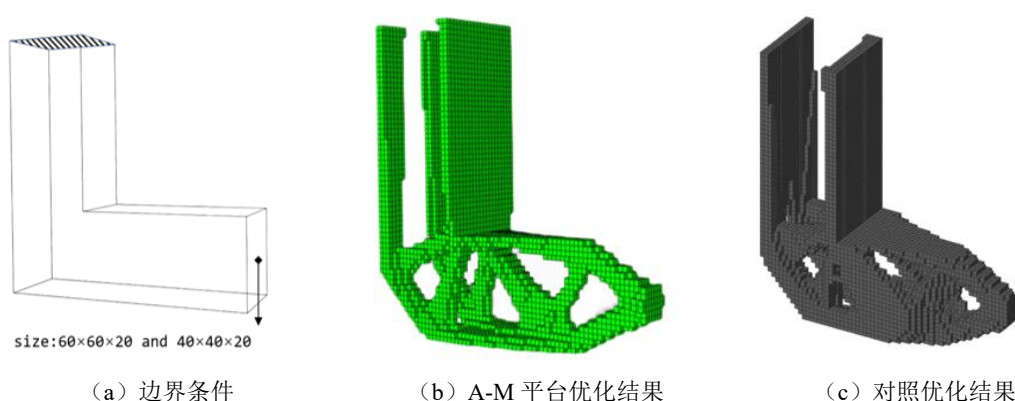


图 3-7 三维 L 型梁的拓扑优化示意图

通过上述二维固支梁和三维 L 型梁的两个拓扑优化结果可知, 拓扑优化结构在体积约束下外形合理、结构美观, 与原 SIMP 法的基础代码相应结果的拓扑构型吻合较好, 证明了本文提出的 ABAQUS-MATLAB 集成拓扑优化方法, 能够有效处理二维和三维拓扑优化问题。

3.4.2 数据交互扩展

MATLAB 核心程序和 Python 核心程序结构良好,针对不同的优化要求或者复杂边界条件结构,只需在原核心代码的基础上增加或修改部分语句就可以实现,下面对扩展代码进行介绍。

ABAQUS 自带的 Python 版本为 2.7,可以通过安装外部库进行程序开发。基于 NumPy 和 SciPy 库,Python 语言可以读取和输出后缀为 .mat 的 MATLAB 文件。需要注意的是,NumPy 是 ABAQUS 自带的库,不需要额外安装。与采用 xlwt 库将数据写入 EXCEL 电子表格相比,最新 ABAQUS 版本运用 .mat 文件传递数据无需额外安装 NumPy 库,但需安装 SciPy 库,且过程复杂,因此以 .mat 文件传输的方式作为扩展方法。

由于 xlwt 库只能处理 65536 行数据的限制,因此前文所述两个核心代码无法优化单元数超过 65536 的有限元模型。另一方面,有限元网格过滤子函数在每个循环中都要处理大量的计算,导致调用.xls 表格需要时间,所以需要对代码稍作修改以提高计算速度。通过在扩展代码中导入 NumPy 和 SciPy 程序库,以 .mat 文件作为 ABAQUS 和 MATLAB 之间的媒介,并在此基础上改变灵敏度滤波子函数的结构,以解决上述两个问题。MATLAB 核心代码需要修改网格过滤子程序,Python 核心代码则需要修改 `get_data` 子程序。修改部分如附录 C 中表 C-1 所示。

在 `get_data` 子程序中 SciPy 被导入并重命名为 `scio`。采用 `scio` 中的 `loadmat` 方法来打开 .mat 文件,该文件中的数据类型是一个矩阵,行数固定为 30,000,列数根据单元总数动态生成,超过最大单元数的部分用 NaN 填充。用于遍历文件的索引 `indexL` 和 `indexR`,则由单元数量计算,最后将单元编号与设计变量生成字典结构。

拓展后 MATLAB 核心程序的网格过滤子函数,如表 C-2 所示。拓展的过滤准备子函数只在第一次拓扑优化循环中运行,以获得一个名为 `elCage` 和 `o` 的单元数组。数组 `o` 内有三个单元,存储有限元模型中三个坐标轴的值,每个轴的坐标值以矩阵的形式存储,矩阵大小由总单元数量决定。数组 `elCage` 的大小与坐标矩阵相同,其中每个单元格存储滤波半径内中心单元的相邻单元。灵敏度过滤子函数在读取数组 `o` 和数组 `elCage` 后开始遍历单元号,索引值由单元编号产生,以当前单元为主单元;取数组 `elCage` 主单元相邻单元号的向量,通过遍历该向量中的每一项,并在数组 `o` 中返回坐标值进行计算,并获取新的灵敏度。

3.4.3 多载荷工况扩展

针对多个载荷工况,只需修改和增加一些语句就可以实现多载荷条件下的结构拓扑优化。以图 3-8 (a) 中的模型为例,有限元模型是一个侧面为正方形的悬臂梁,进行网格划分后,其单元数量为 14400,模型左侧完全固定,右侧两端共有两个负载且负载方向相反。为了提取单载荷的有限元信息,有限元模型中建立了三个

分析步,除了最后一个分析步骤为全负载条件,其余的分析步骤则是单个负载条件。



图 3-8 多载荷工况下悬臂梁的拓扑优化示意图

在 99 行 MATLAB 核心程序中,实现多载荷工况的扩展需要对初始化、读取单元应变能量等部分进行修改,修改部分如表 C-3 所示。在原核心程序的第 5 行添加变量 *loadMax*,赋值为 2,代表模型中共有两个载荷。在第 8 行增加变量 *loadMax* 作为传递给 ABAQUS 的模型数据之一。替换原程序的第 33 至 37 行,其功能是将.xls 文件内每个工作簿上的单元应变能量转换为单元灵敏度,并将同一单元不同分析步中的灵敏度相加;将每个分析步骤中的外力功转换为目标函数并累加。

而 Python 核心程序需要在初始化和单元应变输出等部分进行修改,修改部分如表 C-4 所示。在核心代码的基础程序的第 11 行和第 12 行,增加了变量 *load* 和 *loadMax*,用以存储从 MATLAB 传递的载荷数量。采用新代码代换原代码的第 65-73 行,将每个分析步中计算的单元应变能和外力功,写入.xls 文件的不同工作表。

3.4.4 装配体结构扩展

本集成方法可以借助 ABAQUS/CAE 创建和导入任意形状的几何模型,设计区域的形状限制少,通过对代码的简单修改,就可以将装配体中的零件指定为设计区域和非设计区域进行优化。当处理含有孔的设计域时,在网格划分阶段,只需要确保单元类型为六面体。图 3-9 给出了带孔悬臂梁拓扑优化算例,其中的单元具有不同的尺寸,并以无序的方式排列。

当设计域是装配体中的一个部件时,需要保证该部件在 ABAQUS/CAE 中的命名默认为'Part-1',同时按照表 C-5 中的程序对原核心程序进行修改。Python 核心程序在第 6 行导入 *regionToolset* 模块,在第 33-38 行替换原代码,指定装配体中 Part-1 零件的所有元素为目标区域,修改第 61 行的代码,以输出 Part-1 单元的应变能量。

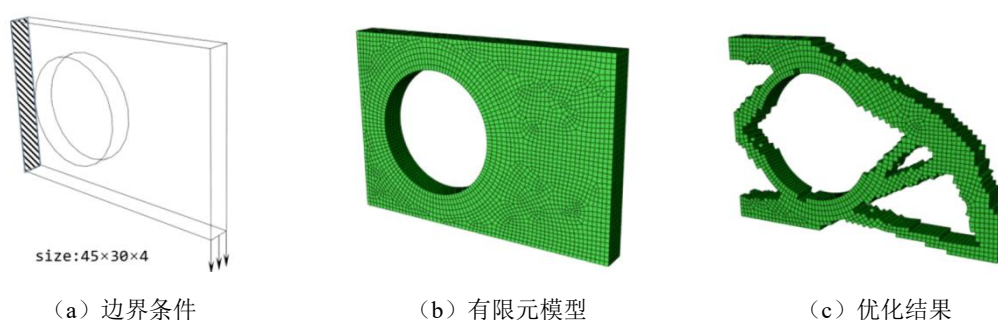


图 3-9 含孔洞悬臂梁的拓扑优化

图 3-10 展示了轮毂装配结构的拓扑优化实例。如图 3-10 (a) 所示, 绿色部分为设计区域, 红色和白色部分为非设计区域, 各部分之间采用 Tie 约束, 内环内表面完全固定, 两对集中力作用于外环。过滤半径为 2, 体积分数为 20%, 最终优化结果, 如图 3-10 (c) 所示。

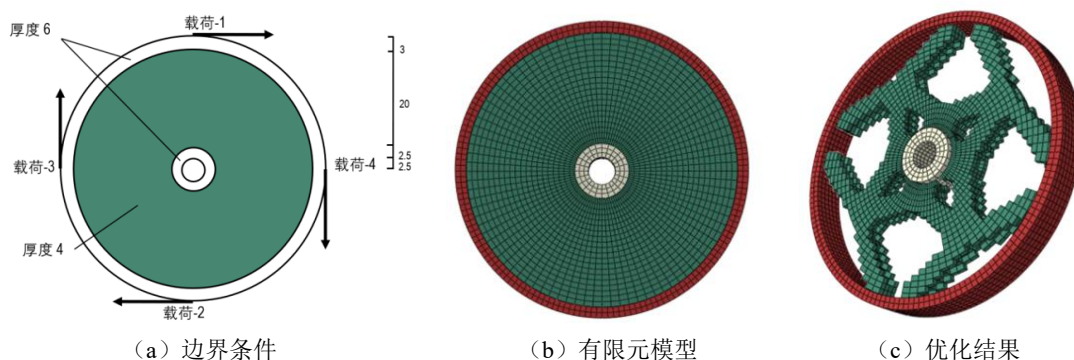


图 3-10 轮毂装配结构的拓扑优化

3.5 本章小结

本章介绍了基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成实现方法, 给出了修改后的 99 行 MATLAB 核心代码和 88 行 Python 核心代码, 并进行了详细解释。使用该集成方法对经典的二维和三维算例进行拓扑优化, 所得优化结果与相同条件下的对照结果构型吻合较好, 证明了该集成方法的有效性。通过对两个核心代码的内容进行扩展后, 该拓扑优化集成方法能够对大规模有限元模型、多载荷工况和装配体中部件进行拓扑优化。该拓扑优化集成方法, 可为众多拓扑优化基础代码与商业有限元软件的拓扑集成提供参考。

第四章 基于等效静态载荷法的拓扑优化集成方法及应用

4.1 引言

众多公开的拓扑优化基础程序往往只能进行静态载荷下的结构拓扑优化，然而实际结构往往承受冲击碰撞等动态载荷的作用。直接对动态载荷下的结构进行拓扑优化，存在计算方式繁琐、计算量大且难以收敛的问题^[64]。等效静态载荷法是四种处理动态拓扑优化问题的方法之一，该方法将动态拓扑优化问题分为：动态碰撞接触分析、计算等效静态载荷、静态拓扑优化、更新设计变量等四个部分，采用线性灵敏度代替了复杂的非线性灵敏度求解，将复杂非线性动态响应拓扑优化问题转化为静态拓扑优化问题。

本章首先进行 BESO 法在 ABAQUS-MATLAB 平台实现的研究，将该平台与采用 ASI 的 BESO 法基础代码进行运算效率对比。而后，实现等效静态载荷法在 ABAQUS-MATLAB 平台运行的集成方法，并对其进行验证。基于此，对 AL/CFRP 吸能盒和单材料吸能盒进行以刚度最大化为目标的碰撞拓扑优化研究，并对对比分析拓扑优化结果。

4.2 等效静态载荷法与 BESO 法拓扑优化理论

4.2.1 等效静态载荷法

采用静载荷方法进行碰撞分析需要创建两个有限元模型，分别作为分析域和设计域，等效静载荷起到串联分析域和设计域的作用。在分析域，执行非线性动态分析，产生动态响应结果，并生成等效静载荷。在设计域，等效静载荷作为加载条件，将动态非线性问题转化为线性静态多工况的优化问题。

在分析域，执行非线性动态碰撞分析时，有限元系统方程可表述为：

$$M(x)\ddot{u}_N(t) + C(x)\dot{u}_N(t) + K(x)u_N(t) = F(t) \quad (4.1)$$

其中， x 是单元的设计变量； M 、 C 、 K 分别表示质量矩阵、阻尼矩阵、非线性刚度矩阵；下标 N 代表该响应是非线性的； \ddot{u} 、 \dot{u} 和 u 分别代表 t 时刻单元的加速度、速度和位移向量； $F(t)$ 是时间步为 t 时的外部负载向量。通过式（4.1）获得 $u_N(t)$ 为 t 时刻的位移长度，即可求等效载荷，等效静载荷的定义可描述为：

$$f_{eq}(s) = K_L(x)u_N(t_s); s = 1, \dots, l \quad (4.2)$$

其中， t_s 是式（4.2）积分过程中的第 s 时间步； l 是时间域的总步长；下标 L 表示线性静态分析， K_L 为线性刚度矩阵。将线性刚度矩阵与式（4.2）求得的节点位移向量相乘，即可得到等效静载荷。若将式（4.2）中的静载荷作为外部载荷，则有：

$$K_L(x)u_L(s) = f_{eq}(s); s = 1, \dots, l \quad (4.3)$$

因此，在任意时刻式 (4.3) 中的节点位移向量 u_L 与式 (4.1) 中的非线性动态节点位移向量 $u_N(t)$ 相同，两者的关系如图 4-1 所示。

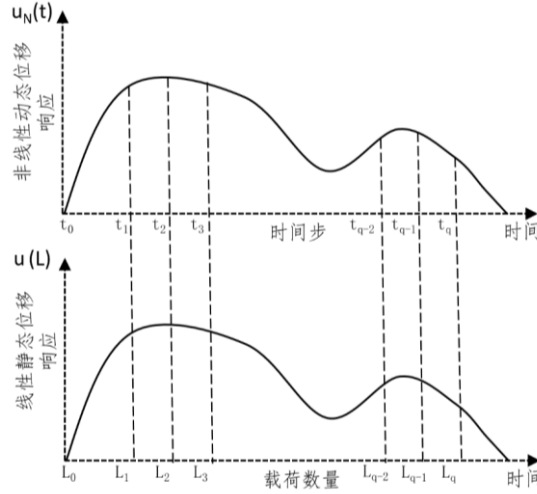


图 4-1 每个时间步动态载荷位移响应与等效载荷下的静态位移响应

4.2.2 BESO 拓扑优化方法

使用 BESO 方法对多个静载荷下的设计域进行拓扑优化。针对体积约束下柔度最小化或刚度最大化，该问题可以表述为：

$$\begin{aligned}
 & \text{find} : x_e = 1, \dots, N \\
 & \min : c = \sum_{j=1}^q w_j F_{eq}(L_j)^T U(L_j), j = 1, \dots, q \\
 & \text{s.t.} : (\sum_{e=1}^N x_e^p K_e) U(L_j) = F_{eq}(L_j) \\
 & \quad \sum_{e=1}^N v_e x_e \leq V^* \\
 & \quad x_e = 1 \text{ or } x_{\min}
 \end{aligned} \tag{4.4}$$

此处， c 为目标函数； x_e 是第 e 单元的相对密度，实体的单元取值为 1，空单元取值为 x_{\min} ； N 是单元的总数； F_{eq} 和 U 分别代表全局力向量和位移向量； K_e 为全局刚度矩阵； p 为惩罚因子； w_j 为相关的权重计算因子；所有单元的体积 v_e 相加得到设计域的总容积； V^* 是体积约束值，它控制着优化目标的体积。

在静态拓扑优化中，采用伴随法计算灵敏度，对于柔度最小化问题，式 (4.4) 中的目标函数可写为：

$$\min : C(x) = F_{eq}(L)^T U(L) = -U^T ((\sum_{e=1}^N x_e^p K_e) U(L) - F_{eq}(L)) \tag{4.5}$$

其中， \tilde{U} 代表任一实数向量， $C(x)$ 是等效静载荷下的目标函数即柔度值。因此柔度值对单元设计变量的导数可以表示为：

$$\frac{\partial C(x_e)}{\partial x_e} = (F_{eq}(L)^T - U^T (\sum_{e=1}^N x_e K_e)) \frac{\partial U(L)}{\partial x_e} = -U^T \frac{\partial K_e}{\partial x_e} U(L) \tag{4.6}$$

在实际有限元分析中 $\tilde{U}=U$ ，因此在柔度最小化问题中单元的灵敏度可以简化为：

$$\frac{\partial C(x_e)}{\partial x_e} = -px_e^{p-1}U^TK_eU \quad (4.7)$$

4.3 基于等效载荷法的 ABAQUS-MATLAB 动态拓扑优化集成方法

4.3.1 基于 BESO 法的 ABAQUS-MATLAB 拓扑优化集成方法

BESO 方法是在渐进结构优化方法 (ESO) 的基础上发展而来。与 SIMP 法相比，两种方法都是基于有限元法的拓扑优化方法，不同点在于双向渐进结构优化方法所得到的结果是不含灰度的。即双向渐进结构优化方法的设计变量不是从 0 到 1 连续变化的，而是只有表示实体单元的 1 和表示空腔的 0。上一章实现了基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成方法，这一方法在运行过程中最耗费时间的过程是材料更新子程序，原因在于采用 SIMP 法，需在 ABAQUS 中创建和设计变量数相同的材料属性，且随着单元数的增多要创建的材料属性就越多。而 BESO 法采用二元设计变量，只需创建两个材料属性而且不受设计域中单元数量的限制，具有收敛快和结果清晰的特点。

根据 ABAQUS-MATLAB 拓扑优化集成框架，实现基于 BESO 法的拓扑优化集成方法，其方式与第三章中实现基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成方法相似。但是由于两种拓扑优化理论的差异和优化目标的不同，需要对 99 行 MATLAB 核心程序中的灵敏度组装、收敛准则和设计变量更新子程序等部分进行修改。根据式 (4.6) 修改后的程序如下：

表 4.1 MATLAB 程序灵敏度组装修改部分

灵敏度组装部分替换代码
<pre>1.ELSEx = xlsread(['ESEDENandALLWKandALLSE.xls'],1); 2.ELSE = ELSEx(:,2)*2; 3.c(loop) = sum(sum((x.*E0).*ELSE)); 4.dc = (x.*E0).*ELSE;</pre>

表 4.2 MATLAB 程序收敛准则修改部分

收敛准则部分替换代码
<pre>1.if loop > 1; dc = (dcn+olddc)/2;end 2.if loop > 10; change = abs(sum(c(loop-9:loop-5))-sum(c(loop-4:loop)))/sum(c(loop-4:loop));end</pre>

表 4.3 MATLAB 程序设计变量更新修改部分

设计变量更新规则替换代码
<pre>1 l1 = min(dc(:)); l2 = max(dc(:)); 2 while (l2-l1)/(l1+l2) > 1.0e-9 3 th = (l1+l2)/2.0;</pre>

```

4.      x = max(0.001,sign(dc-th));
5.      if mean(x(:))-vol > 0
6.          l1=th;
7.      else
8.          l2=th;

```

附录 B 给出的 Python 核心程序,可以根据设计变量的数量自行创建材料属性,因此当只有两个值的设计变量时,Python 核心程序创建一个代表实体的材料属性,另一个代表空腔的材料属性。故只需将控制有限元分析的 Python 核心程序置于同一工作目录中,根据拓扑优化需要调整相关的参数即可。

采用 BESO 法在 ABAQUS-MATLAB 平台进行拓扑优化,计算的三维悬臂梁算例的拓扑优化结果,如图 4-2 所示。拓扑优化构型与 BESO 法的基础代码相应结果吻合较好,表明了集成方法的有效性。

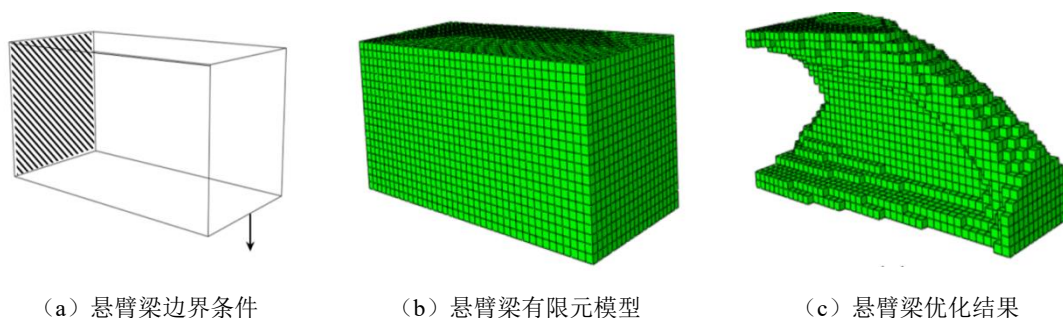


图 4-2 在 ABAQUS-MATLAB 平台中使用 BESO 法对三维悬臂梁的拓扑优化

4.3.2 BESO 法在不同平台运行效率对比

为验证采用本文的集成方法后会提升 BESO 法运行效率,将 A-M 本平台与采用 ABAQUS 脚本接口的 BESO 法^[42]进行对比验证。创建长为 80mm、宽为 20mm 和厚度为 5mm 的悬臂梁,网格单元的尺寸为 1mm;优化目标体积分数为 0.5,过滤半径选择 2.0 和 1.5。在相同的计算机硬件条件下,分别在 ABAQUS 平台和 A-M 平台运行 BESO 法。

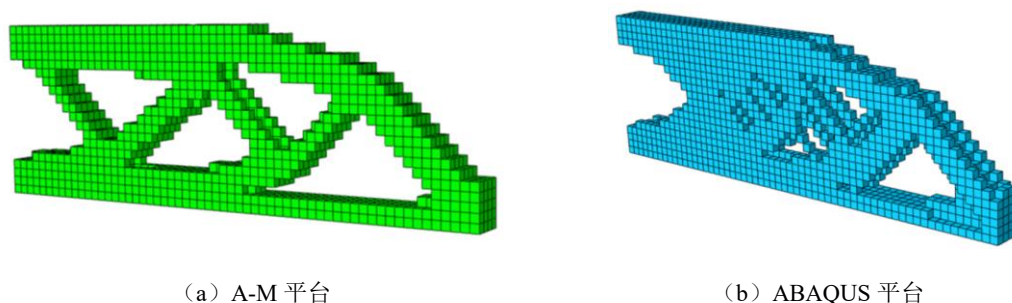


图 4-3 BESO 法在不同优化平台中运行得到的优化结果

图 4-3 给出了过滤半径为 2 时的拓扑优化结果,从图中可知,两个平台的拓扑构型整体相似。表 4.4 给出了过滤半径为 1.5 和 2.0 时拓扑优化用时对比,从表 4.4

可知，当过滤半径为 1.5 时，ABAQUS 单平台的用时为 ABAQUS-MATLAB 平台的两倍以上，其中在网格过滤准备阶段耗费的时间最多为 1032 秒。相比之下 ABAQUS-MATLAB 平台在网格过滤时仅用 15.2 秒，约为 ABAQUS 平台的六十八分之一。其原因是 Python 语言在进行矩阵运算时会遍历矩阵中的每一项，导致 BESO 法的 ABAQUS 平台运算耗时较长。而 ABAQUS-MATLAB 集成方法将网格过滤的准备工作交由 MATLAB 完成，得益于 MATLAB 强大的矩阵运算功能，使得优化耗时较少。尽管 ABAQUS 和 MATLAB 之间的数据交换会占用运行时间，但与 ABAQUS 单平台的计算效率相比数据交换的时间极短。ABAQUS 单平台与 ABAQUS-MATLAB 平台的优化用时差距，还会随着过滤半径的增加而增加，当过滤半径为 2 时，ABAQUS 单平台的总用时约为 ABAQUS-MATLAB 平台的 2.7 倍，大于过滤半径为 1.5 时的 2.4 倍。通过上述比较可见，本文建立的 ABAQUS-MATLAB 拓扑优化集成框架的运行效率更高。

表 4.4 不同平台运行 BESO 用时对比

过滤半径		ABAQUS 单平台	ABAQUS-MATLAB 平台
1.5	总用时	2847 秒	1140 秒
	迭代次数	43 次	44 次
	平均每步用时	66.2 秒	25.9 秒
	网格过滤所需时间	1032 秒	15.2 秒
2.0	总用时	3639 秒	1389 秒
	迭代次数	60 次	53 秒
	平均每步用时	60.42 秒	26.2s
	网格过滤所需时间	2136 秒	17.1 秒

4.3.3 基于等效静载荷法动态拓扑优化集成方法实现

ABAQUS 结果数据库文件中的对象为层级结构。其中，steps 为前处理时创建的分析步，该对象中有一个名为 frames 的对象，功能是存储分析步中所有帧的有限元分析结果。在静力学有限元分析中，用于拓扑优化的分析结果一般取最后一帧数据。在显式动力学有限元分析中，根据工作要求可以定义大量的帧，通过将某一帧内的动态位移响应提取，经计算后可得到该帧对应时间下的等效静态载荷。

采用 4.3.1 中提出的基于 BESO 法的 ABAQUS-MATLAB 拓扑优化集成方法，将动态载荷问题转化为等效静载荷的多工况拓扑优化问题，流程如图 4-4 所示，具体内容如下：

(1)初始化阶段，在 ABAQUS 中创建两个有限元模型，将设计域命名为‘Part-1’。对有限元模型进行网格划分，并记录设计域的网格单元数量。设定工作步的时长以及关键帧的数量。在 MATLAB 中输入优化参数，如设计域最大单元序号、目标体积分数、过滤半径、删除率等。

(2) Python 程序读取设计变量, 在 ABAQUS 中创建材料属性并赋予设计域, 再根据步骤 1 的设定执行显式动力学有限元分析, 生成结果数据库文件。

(3) 读取结果数据库文件, 在内能最大的帧附近取一定数量的帧, 计算每一帧中的等效静态载荷并输出。

(4) 加载静态载荷后进行有限元分析, MATLAB 读取等效静态响应, 按照多载荷工况的拓扑优化方法, 由 BESO 方法求解得到当前体积约束下新的设计变量。

(5) 在设计域体积达到优化目标体积时, 依照 BESO 方法的收敛判断准则, 判断拓扑优化是否收敛, 如果不收敛则返回执行第 2 步。

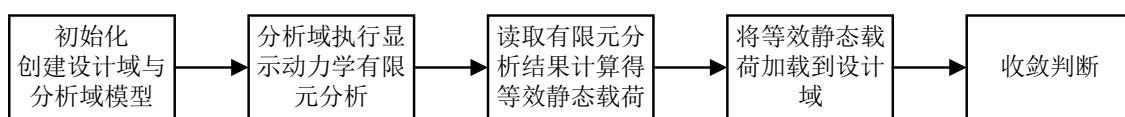


图 4-4 等效静态载荷法拓扑优化集成方法运行流程

4.4 两种吸能盒的刚度拓扑优化

4.4.1 冲击载荷下二维梁结构的拓扑优化

为验证采用集成方法后静态载荷法在 ABAQUS-MATLAB 平台运行的有效性, 对动态载荷拓扑优化的夹紧梁和悬臂梁两个经典算例进行了验证计算。如图 4-5 所示, 夹紧梁的长度为 0.24m、宽为 0.04m、厚度为 0.001m, 两端固定, 中间承受一个最大值为 3kN 的正弦冲击载荷。悬臂梁的长度为 0.08m、宽为 0.04m、厚度为 0.001m, 左端固定, 在悬臂梁上方中点和右下角各施加一个线性冲击载荷, 前者最大值为 2000N、最小值为 -2000N, 后者最大值为 1000N、最小值为 0N。

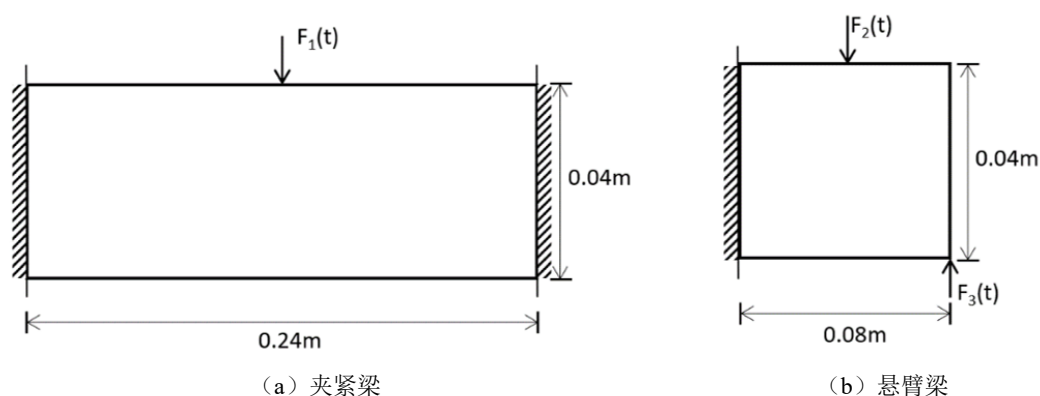


图 4-5 冲击载荷下模型边界条件示意图

两个算例的目标体积均为 0.5, 经过迭代之后得到最终的优化结果, 如图 4-6 所示。与文献^[65]的计算结果进行比较可以看出, 夹紧梁的优化结果吻合比较好; 悬臂梁的拓扑构型与相应结果基本一致, 外部构型吻合较好, 局部略有差异, 其原因可能是网格划分或者优化关键帧数的差异引起的。两个算例造型合理、外形美观,

与对照结果构型趋势相同，证明了本文方法及所建集成方法的有效性。

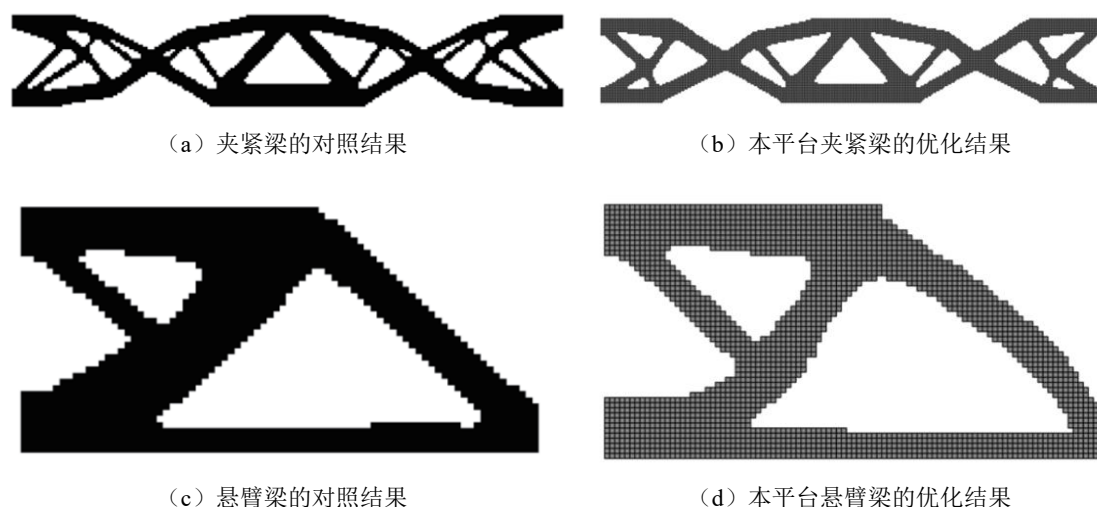


图 4-6 二维结构的动态拓扑优化结果及对比

4.4.2 单材料吸能盒的碰撞模型及拓扑优化结果

吸能盒作为汽车正面碰撞的主要吸能部件，其碰撞时产生的变形对乘员的人身安全以及维修成本有着直接的影响。汽车正面碰撞过程中，碰撞物体与吸能盒之间往往存在角度偏差，吸能盒发生过多的弯曲变形会降低其吸能量效果，从而影响了耐撞性^[59]。

参考汽车修理研究委员会（PCAR）的汽车正面碰撞测试模型^[67]，在 ABAQUS 中建立吸能盒有限元碰撞模型，模型包括刚体墙、吸能盒以及刚体底座等结构，如图 4-7 所示。吸能盒 XY 方向截面长为 0.2m、宽为 0.15m、沿 Z 轴方向的长度为 0.3m。吸能盒材料为铝合金。刚体墙与 XY 平面呈 10° 的偏差，吸能盒与刚性墙间以 15km/h 的相对初速度沿 Z 轴碰撞。刚体底座与吸能盒之间为 Tie 接触，约束底部自由度。

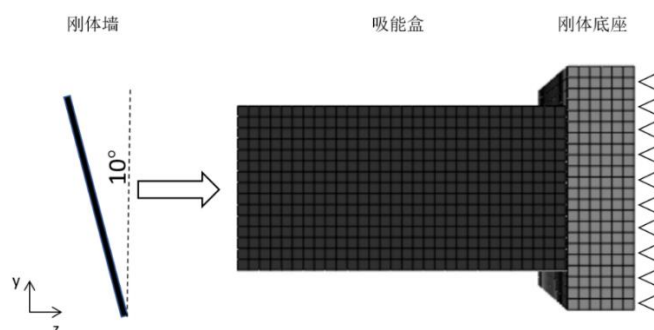


图 4-7 吸能盒的有限元模型

拓扑优化的设计变量为吸能盒中所有单元的材料属性，同时考虑沿挤压方向的约束，故同一 Z 坐标上单元的设计变量相同。为得到清晰的拓扑构型同时达到

减轻重量的目的,将体积分数设为 0.6,以应变能最小化为目标函数进行拓扑优化。

经过 21 次迭代得到最终的优化结果,应变能随迭代次数的变化如图 4-8 所示。在前 10 次迭代中吸能盒材料快速删减,应变能呈现明显的下降趋势,随着删除率到达 0.4 应变能也趋于稳定,即吸能盒的刚度达到最大。

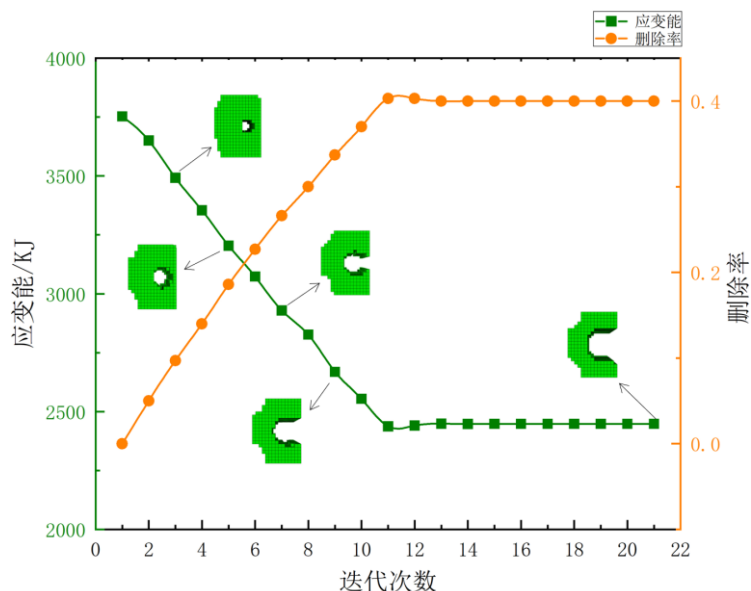


图 4-8 应变能与删除率随迭代次数变化图

图 4-9 给出了铝合金单材料吸能盒碰撞拓扑优化最终结果的斜视图与截面视图。A 侧是吸能盒与刚体墙最先接触的一侧,在两个转角处留有材料以保持结构的稳定并承受刚体墙的撞击,中间部分的材料则被删除。由于 B 侧接触到刚体墙时吸能盒已经产生向 A 侧的变形,所以 B 侧中间部分大部分材料得以保留以支撑刚体墙的撞击,让吸能盒沿撞击方向变形减少偏移的产生。

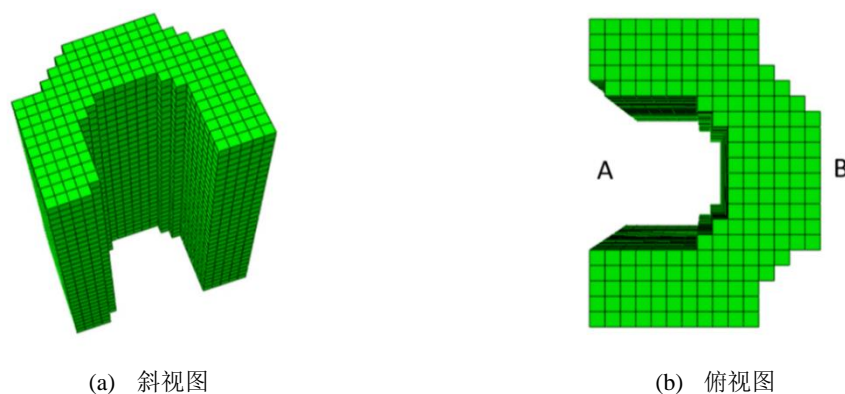


图 4-9 单材料吸能盒拓扑优化结果

4. 4. 3 Al/CFRP 混合结构吸能盒的优化结果及分析

在去除材料追求结构轻量化的同时,结构的性能也会受到影响。为平衡吸能结

构的轻量化与碰撞性能，近年来一些新的高性能轻质材料和结构被应用到吸能结构的开发中，例如定制轧制毛坯、辅助结构、多孔材料和复合材料等。上述材料中碳纤维增强塑料(CFRP)作为一种复合材料，具有独特的机械性能和设计灵活性，被广泛用于汽车工业的结构设计中。

考虑到CFRP机械性能优异且质量轻盈，同时兼顾材料成本，设计了一种CFRP外壳和铝合金内芯组成Al/CFRP混合结构的吸能盒，如图4-10所示。为优化该混合吸能盒的材料布局，采用前节建立的ABAQUS-MATLAB碰撞拓扑优化集成方法，对该混合结构吸能盒进行动态碰撞拓扑优化分析。

在原有单材料吸能盒有限元模型的基础上，创建一长宽高与吸能盒相同的薄壁，CFRP材料为T300^[67]。吸能盒复合材料铺层数为8层，每层厚度为0.125mm，总厚度为1mm铺层角度依次为 $[0^\circ / \pm 45^\circ / 90^\circ / 90^\circ / \pm 45^\circ / 0^\circ]$ 。将复合材料外壳以Tie约束的形式与铝合金内芯建立连接关系。

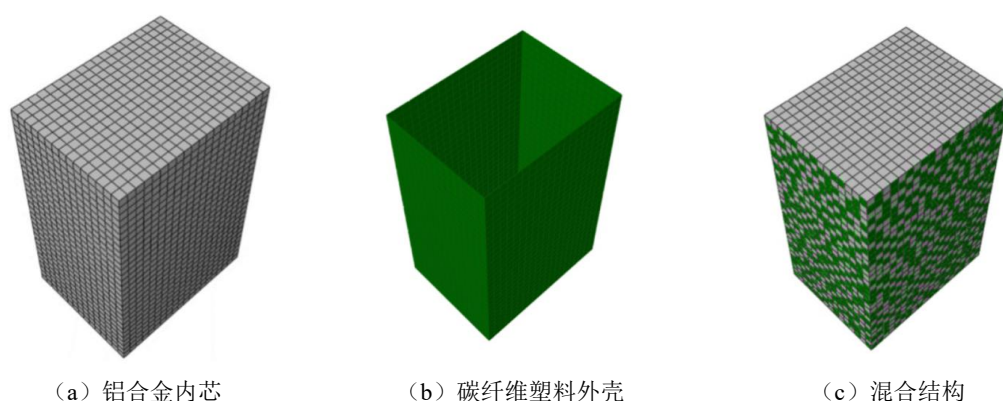


图 4-10 Al/CFRP 混合结构吸能盒有限元模型

混合结构吸能盒选定铝合金内芯为设计域，碳纤维外壳为非设计域。设计域的单元数与单一材料吸能盒的单元数一样，为 9000 个单元，优化参数也保持一致。考虑材料和接触非线性对 Al/CFRP 混合结构吸能盒碰撞进行拓扑优化。

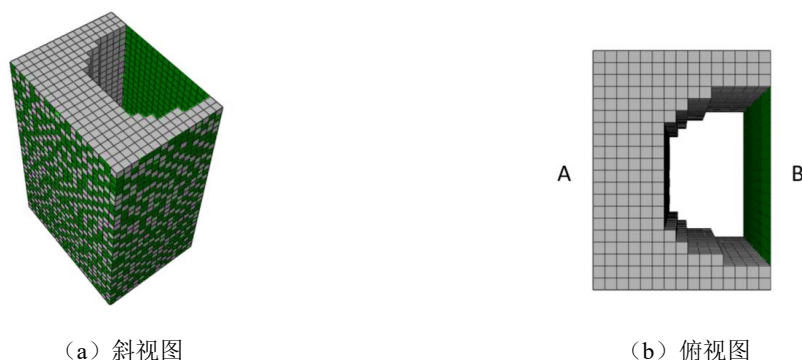


图 4-11 Al/CFRP 混合结构吸能盒拓扑优化结果

经过 31 次迭代，得到 Al/CFRP 混合结构吸能盒的优化结果，如图 4-11 所示。

将优化结果与单一材料吸能盒优化结果相比，两个设计域的拓扑构型存在差异。在图 4-11 (b) 中 A 侧为吸能盒与刚体墙先撞击的一侧，为了承受墙体的冲击保持结构的稳定，大部分材料在此堆积。而在 B 侧仅在两个转角留有材料，中间部分的材料则被删除。引起单材料吸能盒与 AL/CFRP 混合结构吸能拓扑优化结果差异的原因主要有二：其一，AL/CFRP 混合结构吸能盒外部的碳纤维增强材料限制了铝合金内芯的变形，减少了上部截面受到冲击后的倾斜程度；其二，碳纤维外壳在 B 侧提供了有效的支撑，使吸能盒整体结构在受到侧面冲击后仍能保持稳定。

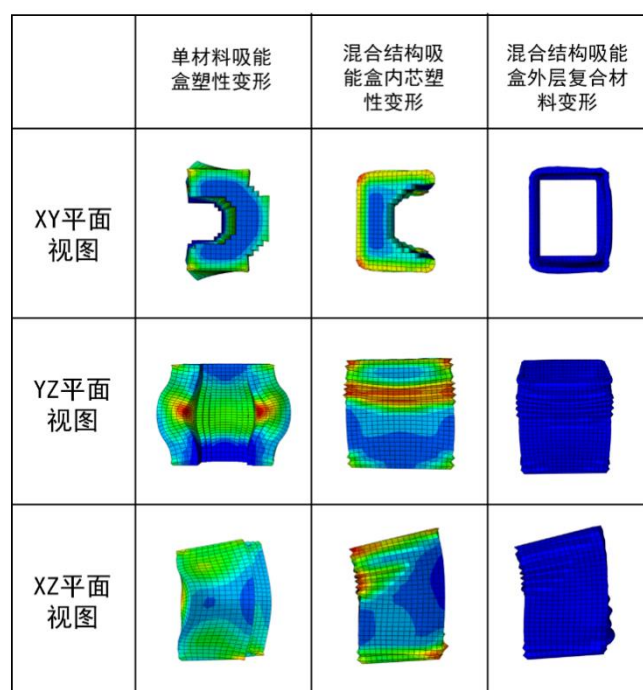


图 4-12 两个优化结果的变形情况

图 4-12 给出了单一材料吸能盒与 AL/CFRP 混合结构吸能盒优化结果的变形情况。由图中可以看出，单材料吸能盒变形幅度较大，中间部位尤为明显，先受到撞击的一侧吸能盒材料有向外胀开的趋势，随后受到撞击的一侧吸能盒呈现大幅度的弯曲。与之相对的，AL/CFRP 混合结构吸能盒内芯的变形程度较小，变形主要集中在内芯上方先受到撞击一侧。在后撞击一侧内芯的上方有一定幅度的弯曲，下方存在小变形。AL/CFRP 混合结构吸能盒的复合材料外层整体变形与内芯相同，在下方内芯空缺处产生了向外的小幅度变形。

图 4-13 给出了两种结构吸能盒的力-位移曲线。可见单一材料吸能盒与 AL/CFRP 混合结构吸能盒优化结果在吸能方面存在差异。从图 4-13 可知，两种吸能盒的力与位移曲线迅速达到峰值，其中单材料吸能盒产生的反力在到达峰值后又迅速的降低，在位移达到 20mm 时不再下降而保持在一定范围内波动。AL/CFRP 混合结构吸能盒产生的反力在达到峰值后，随着位移的增加，在经过小幅度的下降

之后又产生了大幅度的下降，直到位移达到 39mm 时又开始上升，最终在碰撞结束时趋于稳定。力-位移曲线下面积的大小，反映了两种吸能盒吸能量的多少，Al/CFRP 混合结构吸能盒的力-位移曲线下面积要明显大于单材料吸能盒的面积。此外，单材料吸能盒的最终位移为 48.9mm，Al/CFRP 混合结构吸能盒的最终位移为 44.7mm。在汽车的正面结构中还包括散热器和头灯等部件，吸能盒在受到撞击后必然会产生变形，但是如果变形过大则可能损坏到其他部件，增加维修成本。可见，Al/CFRP 混合结构吸能盒比单材料吸能盒对其余部件的保护性更好。

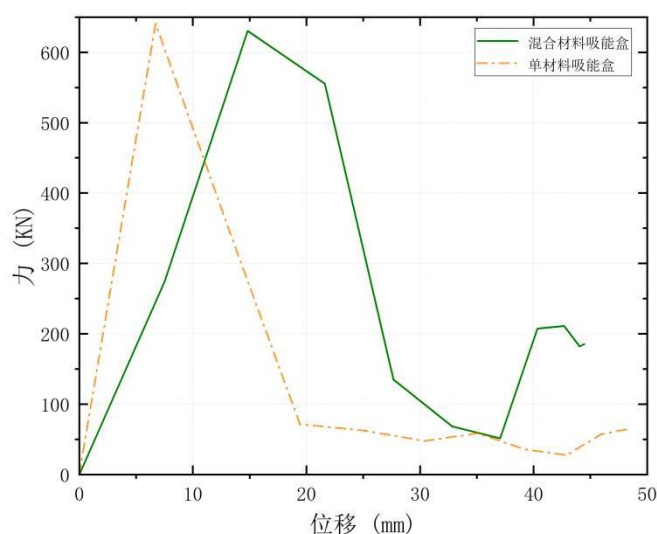


图 4-13 两种吸能盒优化结果的力-位移曲线

通过单材料吸能盒与 Al/CFRP 混合结构吸能盒的优化结果及力-位移曲线的对比分析可知，除了拓扑优化截面形状的差异外，两种吸能盒结构在受到撞击后的性能表现也不同，Al/CFRP 混合结构吸能盒动态优化结果在结构稳定性方面优于单材料吸能盒的动态拓扑优化结果。

4.4.4 不同碰撞角度下 Al/CFRP 混合结构吸能盒的优化结果

为了探究碰撞角度对 Al/CFRP 混合结构吸能盒拓扑优化结果的影响，在上一节刚体墙倾斜 10° 的模型基础上，进行 5° 、 15° 和 20° 三种倾斜角度下的碰撞拓扑优化。有限元碰撞模型仅调整刚体墙的倾斜角度，其余有限元参数和拓扑优化参数，与前面保持一致。

图 4-14 给出了 4 种不同碰撞角度工况的动态拓扑优化结果。其中，工况 1 (5° 倾斜) 与工况 2 (10° 倾斜) 的金属内芯截面形状几乎一致。当倾斜角度超过 10° 后，工况 3 (15° 倾斜) 与工况 2 内芯的截面形状存在较大差异，在工况 3 中材料分部向 A 侧转移，而在 B 侧两个角落位置也没有材料填充。当倾斜角度到达 20° 后，工况 4 (20° 倾斜) 中内芯的材料则全部集中在 A 侧。造成这种现象的原因

是：随着刚体墙倾斜角度的增加，吸能盒先受到撞击的一侧先产生变形，而且倾斜的角度越大吸能盒截面产生的变形也越大，导致未变形的截面不会受到刚体墙的撞击，最终出现了工况 4 中的拓扑优化结果。结果表明，碰撞角度为 5° 和 10° 工况下的 Al/CFRP 混合结构吸能盒，其动态拓扑优化结果的截面形状更为理想，可指导该混合结构吸能盒的结构设计。

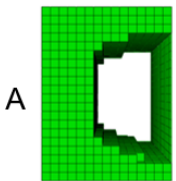
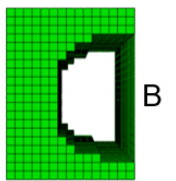
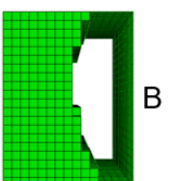
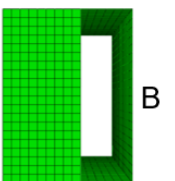
工况编号	1	2	3	4
优化结果				
墙体倾斜角度	5°	10°	15°	20°

图 4-14 不同撞击角度下吸能盒的拓扑优化结果

4.5 本章小结

本章在基于 SIMP 法拓扑优化集成方法的基础上，通过对 MATLAB 核心程序作略微修改，将优化算法改为 BESO 法，实现了基于 BESO 法的静态载荷拓扑优化集成方法，表明 ABAQUS-MATLAB 拓扑优化集成框架拓展性较好。将 BESO 法在 ABAQUS-MATLAB 平台和 ABAQUS 单平台的运算效率进行对比，表明当过滤半径为 1.5 时，前者比后者快 2.4 倍，且随着过滤半径的增加两者的差距也会增加，证明了采用了本文的集成方法之后 BESO 法在 ABAQUS-MATLAB 平台运行效率更高。引入等效静态载荷理论，将动态载荷转变为多个静态载荷工况，实现基于 BESO 法的 ABAQUS-MATLAB 碰撞拓扑优化集成方法，并对经典算例进行计算。通过与对照算例的比较，证明了等效静态载荷在 ABAQUS-MATLAB 平台的集成方法的有效性。

采用上述碰撞拓扑优化集成方法，对单一材料吸能盒和 Al/CFRP 混合结构吸能盒进行了动态拓扑优化，优化的目标是结构的应变能最小，保证优化结果的刚度最大，以防止在碰撞过程中吸能盒产生过度的变形。通过 Al/CFRP 混合结构吸能盒拓扑优化结果与单材料吸能盒的优化结果对比发现，两个结果除了在截面形状上有明显的差异外，混合结构吸能盒的优化结果在结构稳定性和吸能方面，均优于单材料吸能盒。通过 Al/CFRP 混合结构吸能盒在不同倾斜角度工况下的碰撞拓扑优化结果对比分析发现，小角度碰撞下 Al/CFRP 混合结构吸能盒的碰撞拓扑优化

结果更为理想。

第五章 基于混合元胞自动机法耐撞性拓扑优化集成方法及应用

5.1 引言

交通运输、航空航天等领域的对其装备的耐撞性具有很高的要求，而拓扑优化方法因其能在概念设计阶段提升产品耐撞性，在产品耐撞性设计中得到了广泛应用。混合元胞自动机法（HCA）是动态拓扑优化的主要方法之一，与等效静态载荷法需要和静载荷拓扑优化方法一起进行动态拓扑优化不同，HCA 法是采用非灵敏度的拓扑优化技术，能直接对碰撞系统进行动态拓扑优化，无需结合其他的拓扑优化方法，受到了耐撞性拓扑优化研究学者的高度重视。此外，HCA 法还具有收敛速度快和易于二次开发等特点。

本章对混合元胞自动机方法的原理进行介绍，因该方法不使用灵敏度的特性，无法通过直接修改现有的核心代码实现两个平台的集成，故需研究 HCA 法 ABAQUS-MATLAB 耐撞性拓扑优化集成方法的实现问题。对比分析 HCA 方法不同更新规则 and 不同元胞自动机对耐撞性拓扑优化结果的影响。最后，采用基于混合元胞自动机的 ABAQUS-MATLAB 拓扑优化集成方法以耐撞性为目标对汽车吸能盒进行了拓扑优化研究。

5.2 混合元胞自动机拓扑优化方法

混合元胞自动机拓扑优化方法，是通过结合元胞自动机（CA）范式和基于有限元的结构优化理论而形成的。最初该方法被用于医疗领域的人骨重塑预测^[66]，之后该方法改进后被引入到结构设计领域。改进后的混合元胞自动机法结合了局部更新方案，比如控制规则和比率技术。其中，控制规则的作用是将场变量 y_i 更新至符合优化要求的最佳数值或者设定目标值 y^* 。以 y_i 和 y^* 的差函数表示设计变量 x_i 的变化，即：

$$\begin{aligned} x_i(t+1) &= x_i(t) + f(e_i(t)) \\ e_i(t) &= y_i(t) - y^* \end{aligned} \quad (5.1)$$

$f(e_i(t))$ 为开-关控制类型，可表示为：

$$f(e_i(t)) = \begin{cases} +c_T, & \text{if } e_i(t) > 0 \\ 0, & \text{if } e_i(t) = 0 \\ -c_T, & \text{if } e_i(t) < 0 \end{cases} \quad (5.2)$$

此处， c_T 的值是位于 0 到 1 之间的常数。

若以比例-积分-微分控制器（PID）描述，该 $f(e_i(t))$ 表示为：

$$f(e_i(t)) = c_p e_i^k + c_I [e_i(0) + e_i(1) + \dots + e_i(t)] + c_D [e_i(t) - e_i(t-1)] \quad (5.3)$$

式中, c_p 、 c_I 和 c_D 均为正标量, 分别代表比例增益、积分增益和倒数增益。大量研究成果验证了式(5.2)和式(5.3)两种控制规则的收敛性, 被广泛用于混合元胞自动机拓扑优化研究中。

5.2.1 混合元胞自动机的组成

混合元胞自动机模型包括: 网格单元、每个网格单元的有限元状态变量、与每个状态相关的一组规则等 3 部分。对于拓扑优化问题, 可以定义二维和三维的网格单元, 对于每个网格单元其相关联的状态向量 α_i 可以表示为:

$$\alpha_i = [x_i, y_i] \quad (5.4)$$

其中, x_i 是设计变量的集合, y_i 是第 i 个单元格状态变量的集合。对与每个状态, 都有相应的更新规则来定义其在时间(迭代次数)上的演变。在混合元胞自动机法中, 设计变量在优化阶段被更新, 比如在局部控制规则中被更新, 而在有限元分析阶段则只更新状态变量。更新规则适用于所有的网格单元而且不依赖单元格的位置。

更新规则需要使用到每个单元附近的邻域信息。对于单元的大小和位置一般没有限制, 但是如果使用了某种自动机, 邻域就必须保证设计域中每个单元的自动机邻域形状是一样的。图 5-1 和图 5-2 展示了二维和三维设计中常用的自动机邻域形状。

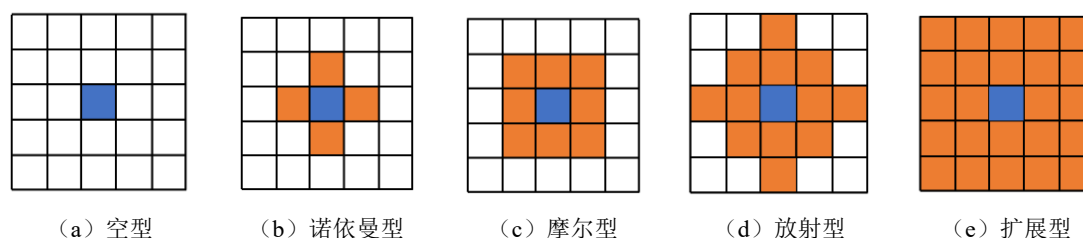


图 5-1 二维结构中不同形状的自动机邻域

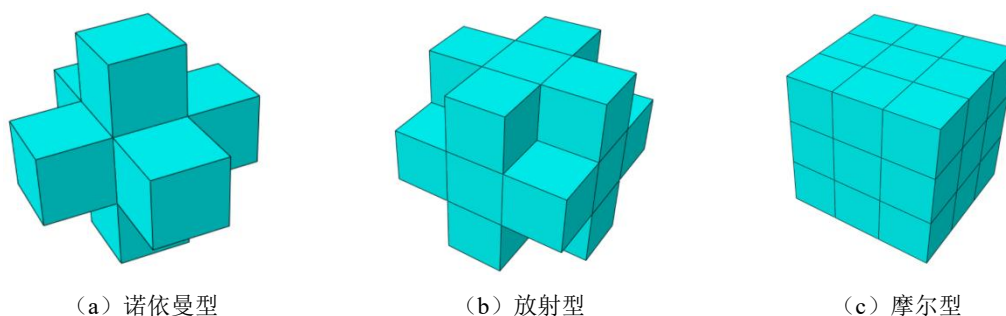


图 5-2 三维结构中不同形状的自动机邻域

由于元胞自动机邻域形状是固定的, 当单元处于设计域的边界时, 为了满足收

集邻域中单元信息的条件,处于边界外的邻域单元根据设计要求添加。边界外的单元添加形式共有四种,分别是:固定型、绝热型、反射型和周期型。对于固定型,处于边界外的邻域单元将利用预先设定的固定状态单元进行填充;对于绝热型,处于边界外的邻域单元将采用中心单元状态值进行填充;而对于反射型,则边界外的邻域单元通过复制对称位置单元的状态信息获得;若设计域被包裹成环形,则需要采用周期型边界条件。上述添加规则,如图 5-3 所示。本文采用是固定边界条件,邻域边界外的额外单元,采用没有物理和机械特性的空白数值。

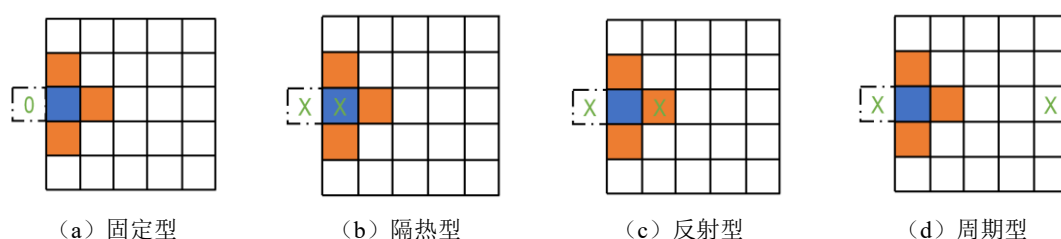


图 5-3 不同的元胞自动机边界添加形式

5.2.2 全局和局部控制策略

耐撞性设计的优化目标是使结构在碰撞过程中最大限度地吸收能量。为防止结构失效,需设置应变约束,使结构的最大塑性应变需在材料强度极限内。

基于满应力设计思想,学者们采用 HCA 法针对非线性瞬态问题进行了耐撞性设计。该方法将每个单元的内能密度更新至指定的目标数值或者设定数值,利用合适的优化算法,通过改变单元的相对密度来循环迭代更新单元刚度。需要解决的结构问题可以表述为找到满足局部和全局约束的材料最优分布。该问题的全局公式如下:

$$\begin{aligned}
 & \text{find } x \\
 & \text{s.t. } h(x)=0 \\
 & \quad g(x) \leq 0 \\
 & \quad H(x)=0 \\
 & \quad G(x) < 0 \\
 & \quad x_i \in \{0,1\}, i=1,\dots,n
 \end{aligned} \tag{5.5}$$

其中, h 和 g 是局部等式和不等式约束的集合, H 和 G 是全局等式和不等式约束的集合, x 是设计变量的集合,设计变量 x_i 代表材料存在或者不存在。上式对局部约束比较宽松,其设计变量可以在 0 和 1 之间取值。

局部控制规则是一种求解材料在每个单元中分布的方法,该规则只对设计变量 x_i 进行更新。为了在碰撞过程中,使混合元胞自动机方法找到最优的材料布局,局部控制策略可以描述为:

$$\begin{aligned}
& \text{find } x_i \\
& \text{s.t. } y_i(x_i) - y^* = 0 \\
& x_i \in \{0,1\}
\end{aligned} \tag{5.6}$$

其中, y_i 是状态变量, y^* 是目标数值。对于二元设计变量的情形, 因等式约束可能不满足, 可采用如下松弛公式:

$$\begin{aligned}
& \min |y_i(x_i) - y^*| \\
& \text{s.t. } 0 \leq x_i \leq 1
\end{aligned} \tag{5.7}$$

5.2.3 更新规则

局部控制规则按照如下表达式更新设计变量:

$$x_i(t+1) = \bar{x}_i(t) + f(\bar{y}_i(t)) \tag{5.8}$$

式中, t 代表离散时间步或者迭代次数, \bar{x}_i 和 \bar{y}_i 分别代表设计变量和场变量的有效值, 这些有效值可以表示为:

$$\bar{x}_i(t) = \frac{x_i(t) + \sum_{j=1}^N x_j(t)}{N+1} \tag{5.9}$$

$$\bar{y}_i(t) = \frac{y_i(t) + \sum_{j=1}^N y_j(t)}{N+1} \tag{5.10}$$

其中, 索引 j 表示相邻的单元格。如果采用比例控制法, 则式 (5.8) 中的 $f(\bar{y}_i(t))$ 可以定义为:

$$f(\bar{y}_i(t)) = c_p \times (\bar{y}_i(t) - y^*(t)) \tag{5.11}$$

其中, $y^*(t)$ 是场变量 $y_i(t)$ 的设定点, c_p 是比例增益常数。在开-关、导数、积分和比例更新规则中, 本文采用比例更新规则式 (5.11) 进行设计变量的更新。

5.3 基于混合元胞自动机法的拓扑优化集成方法实现

与 SIMP 方法和 BESO 方法在 ABAQUS-MATLAB 拓扑优化集成方法的实现方式相似, 混合元胞自动机法同样基于 ABAQUS-MATLAB 拓扑优化集成框架, 采用 MATLAB 核心程序和 Python 核心程序控制。由于混合元胞自动机法和 SIMP 法在原理上存在较大差异, 无法直接利用已有的核心程序, 需重新编写核心程序, 以实现混合元胞自动机法的耐撞性拓扑优化集成方法。本节按照程序运行顺序对附录 D 中的 MATLAB 核心程序, 以及附录 E 中的 Python 核心程序进行详细阐述。

5.3.1 初始化

初始化阶段, 沿用 topMA 核心程序方法, 例如输入与导出模型信息和导出循环信息, 但需对优化参数和优化变量的初始化部分进行修改。根据混合元胞自动机方法理论, maHCA 程序只需模型最大单元数和目标体积分两个参数, 即

$maxElNum$ 和 $tMass$ (第 1 行)。创建 $xPro$ 、 $iedPro$ 和 $loopMax$ 等三个变量, 前两个变量分别是长度与设计变量相等的零向量, 用以储存过滤后的设计变量和内能, $loopMax$ 则是循环次数的最大限定值(第 16-18 行)。初始的 $change$ 变量取无限大, 当该变量大于收敛标准或迭代次数小于迭代次数上限时开始循环(第 19-21 行)。在循环之初, 与 $topMA$ 核心程序类似, 输出循环信息与设计变量后调用名为 $SrEaba_out$ 的 Python 核心程序。

5.3.2 有限元分析前处理

基于 HCA 法耐撞性拓扑优化集成方法中的 Python 核心程序, 采用了与 SIMP 法的 Python 核心程序相同的初始化和设计变量子函数代码。因此, 附录 E 中只给出本章 Python 核心程序的主函数部分。

进入主函数, 定位部件 ‘Part-1’ 为设计区域, 把部件中的单元信息和节点信息传递给 $eleee$ 变量和 $nodee$ 变量(第 1-8 行)。将设计域中的所有单元加入到名为 $mySet$ 的集合并传递给变量 $targetRegion$ (第 9-11 行)。执行 get_data 设计变量组装子函数, 得到单元编号与设计变量呈键值对的字典结构 $xCoE$ (第 12 行)。设定惩罚因子与初始杨氏模量, 遍历 $xCoE$ 中所有的键, 为更新单元的材料属性做准备(第 13-17 行)。因碰撞过程中会产生塑性变形, 所以弹性假设不再适用。在对材料属性的弹性参数更新的同时, 还要更新塑性参数。材料模型可表示如下:

$$E_i = x_i^p E_0 \quad (5.12)$$

$$\sigma_i^Y = x_i^p \sigma_0^Y \quad (5.13)$$

$$E_i^{t_k} = x_i^p E_0^{t_k} \quad (5.14)$$

其中, E_i 、 σ_i^Y 和 $E_i^{t_k}$ 分别代表杨氏模量、屈服应力和硬化模量。 x_i 为第 i 个单元的设计变量, p 为惩罚因子。 $penalXe$ 和 $penalXp$ 分别是经过惩罚后的弹性和塑性设计变量(第 18-21 行)。本章所采用铝合金材料属性, 如表 5.1 所示。

表 5.1 铝合金材料属性

属性	数值
密度 (KG/M ³)	2700
弹性模量 (GPA)	70
屈服应力 (MPA)	180
硬化模量 (/MPA)	(0,180), (0.01,190), (0.02,197), (0.05,211.5), (0.1,225.8), (0.15,233.6), (0.2,238.5), (0.4,248.5)

创建 saK 、 svK 和 hK 三个列表, 分别表示初始屈服应变、初始屈服应力和初始硬化模量(第 22-24 行)。根据式 (5.12) 和 (5.13) 对 svK 和 hK 进行更新, 得

到插值后的屈服应力 stK 和硬化模量 hK (第 25-26 行)。更新材料属性编号, 组装屈服应变 (第 27-29 行)。

如图 5-4 所示, HCA 方法通过建立分段线性模型, 近似地描述各单元的塑性行为。由更新后的屈服应力和硬化模量, 得到存储塑性应变的列表 saK (第 30-32 行)。为便于材料更新, 将塑性应变与对应的塑性应力转化为元组类型。创建材料属性, 采用 `Density` 关键字输入密度信息 (第 33-36 行)。

当设计变量过小时, 会使分段线性模型中的应变十分接近, 造成 ABAQUS 显式分析无法进行插值导致程序出错, 所以应变和应力数据需要过滤后才能生成材料属性。其过滤规则为: 当存在两组应变之间的差值小于 10^{-4} 时, 删去数值小的应力与应变, 直到所有的相邻应变之间的差值大于 10^{-4} (第 37-44 行)。采用 `Plastic` 方法将塑性属性添加进材料属性。最后, 由材料属性得到截面属性后赋予设计变量所对应的单元 (第 45-49 行)。

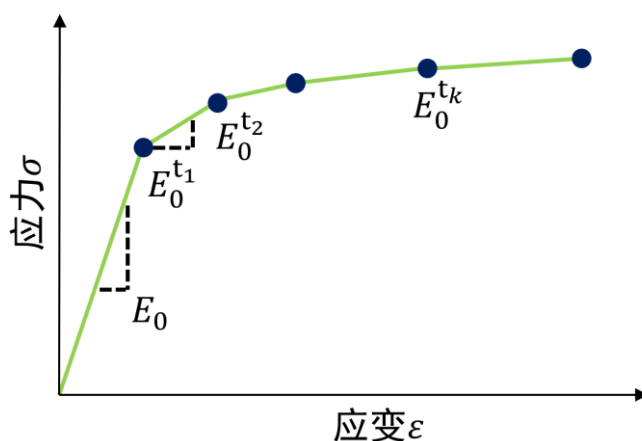


图 5-4 塑性应力应变分段模型

执行有限元分析并等待分析结束 (第 50-52 行)。打开结果数据库文件, 分别采用 `ELSE`、`ELPD` 和 `ELASE` 等关键字, 将弹性应变能、塑性应变能和人工应变能等能量, 输出到指定的 `.mat` 文件后关闭结果数据库文件 (第 53-69 行)。

5.3.3 元胞自动机组建

混合元胞自动机法利用元胞自动机更新单元的有限元属性信息, 无需网格过滤; 另一方面, 在 ABAQUS-MATLAB 拓扑优化集成框架中, 有限元模型由 ABAQUS 处理, 各单元位置信息需要传递到 MATLAB 核心程序。获得每个单元自动机内邻域内单元的编号, 是 Python 核心程序需要实现的重要问题。

以下内容在 Python 核心程序的第一次迭代执行时, 创建空列表 `elNuNd` 和 `ndNuEl` (第 70-72 行)。遍历所有单元, 采用 `getNode` 方法, 返回单元的所有节点并存储到列表 `ndN` 中。利用 `label` 方法提取 `ndN` 中所有节点的编号, 并以列表的形式添加进 `elNuNd` (第 73-78 行)。遍历完成后, 列表 `elNuNd` 中拥有和单元数量一

样多的子列表,其中每个列表都存储着与其索引值相同编号的单元的所有节点。根据同样的思路,遍历所有节点,将共用节点的所有单元的单元编号以列表的形式存储到 *ndNuEl* (第 79-84 行)。

如果设计域中采用 CPS4R 四节点二维单元时,则遍历 *elNuNd* 中的所有子列表。根据子列表中的每个节点编号,在 *ndNuEl* 中提取节点所对应的单元编号,将其存储到列表 *mid* 中。此时 *mid* 中共有四个子列表,子列表的索引值为 0 到 3。故将 0 到 3 之间成对的排列组合方式传递到变量 *m*。根据 *m* 中每种组合方式,对 *mid* 中的子列表求交集。将所得结果添加进变量 *result* 中,这些单元编号即为当前单元的元胞自动机邻域。因存在部分元胞自动机的单元处于设计域之外的情形,导致 *result* 中并不一定全为五个单元编号的列表,所以当 *result* 中的元素数量不足 5 个时,需用某一数值填充,该数值取为最大单元数与 1 的和。遍历所有单元后,得到一个列数为最大单元数、行数为 5 的矩阵,将该矩阵命名为 *cAV* 并输出到 *caValue* 文件 (87-107 行)。如果是 C3D8R 八节点三维单元类型,采用与二维单元类似的方法,输出元胞自动机的邻域信息 (108-131 行)。

5.3.4 有限元分析结果组装

MATLAB 核心程序在 Python 核心程序执行完毕后继续执行。读取存储弹性应变能、塑性应变能、人工应变能和自动机邻域信息的 .mat 文件,将其中的内容递给 *eleRaw*、*pedRaw*、*eaRaw* 和 *cA* 等变量 (第 28-31 行),其中前三个变量被组装成内能 *iedRaw* (第 32 行)。采用固定型添加方式,当自动机邻域中有设计域外的单元时,用 0 值作为单元的信息。因此,在组装元胞自动机之前,对设计变量和内能补充一项 0 值 (第 33-34 行)。当拓扑优化处于第一次循环时,定义初始内能目标即式 (5.1) 中的 y^* 。因内能目标值会随循环而更新,需设定内能的初始值,其值为总内能平均值与体积约束目标值的乘积 (第 35-37 行)。将变量 *x* 传递给变量 *xFc* 准备更新单元的有限元状态。

5.3.5 挤压条件

为获得吸能盒拓扑优化截面形状,约束其挤压方向以尽量获取完整的横截面形状。为此,先读取单元的中心坐标数据,确定挤压方向为 Z 轴方向,获取 Z 坐标值最小 X-Y 平面中的所有单元坐标信息,并将其设为初始层。为初始层中每个单元创建一个向量,采用 *find* 方法将相同 (X,Y) 坐标的单元编号添加进同一向量。将所得到的向量组成一个矩阵,该矩阵即为挤压条件下单元的相互位置信息 (第 97-108 行)。读取单元约束矩阵,将相同 (X,Y) 坐标的单元的内能相加求平均值,将该平均值分配给每个单元,即: $U_{i1}=U_{i2}=U_{i3}=\dots=U_{iN}$,使得挤压方向上每个单元的设计变量趋于相同 (第 39-49 行)。

5.3.6 单元状态更新

根据自动机邻域信息更新每个单元中的设计变量和内能，更新方式为遍历所有单元，按照邻域中所存储的单元编号返回所有单元的设计变量并取和，之后根据单元个数取平均数（第 50-53 行）。内能也按相同的方式进行更新（第 54-56 行）。在拓扑优化过程中，由于求解动态问题具有不稳定性，材料分布会在迭代过程中产生振荡，通过采用对内能的历史信息平均加以解决。创建变量 *iedProOld* 保存上一次迭代的内能，当循环次数大于 2 次时，将上一次迭代与当前迭代中的内能平均得到当前的内能（第 57-61 行）。创建与矩阵 *xPro* 相同尺寸的零矩阵，更新设计变量。

5.3.7 设计变量更新

将所有设计变量相加得到当前设计域的体积分数（第 63-64 行）。在向量 *xPro* 中查找 x_{min} 和 x_{max} 之间的设计变量（第 65 行）。根据式 (5.11) 的比例方法，对设计变量进行更新并传递给变量 *x*，并计算设计域的体积分数（第 66-68 行）。

当前体积分数与目标体积分数相差大于原始体积的 0.005 倍时，执行体积约束内循环（第 72 行）。在内循环中对设计变量再次进行更新，并计算新的体积分数，根据当前体积分数与目标体积分数之间的比值，更新内能目标值。当满足体积约束后，停止内循环（第 73-85 行）。

$$\frac{|\Delta V^{(t)}| + |\Delta V^{(t-1)}|}{2V_0} \leq \varepsilon \quad (5.12)$$

当主循环大于 2 次时，根据式 (5.12) 开始计算收敛差值，其中 $\Delta V^{(t)}$ 是当前迭代中优化前后设计域的差值，记为 *xC*； $\Delta V^{(t-1)}$ 是上一次迭代中优化前后设计域的差值，记为 *xCold*。 V_0 是设计域的初始体积， ε 在程序中设定为 0.001（第 86-93 行）。最后，输出迭代次数、当前体积、目标内能、内能之和和收敛差值。

5.4 吸能盒的耐撞性拓扑优化

为验证所实现的混合元胞自动机 ABAQUS-MATLAB 耐撞性拓扑优化集成方法的有效性，以冲击载荷下的固支梁作为优化算例。该梁的有限元模型和边界条件如图 5-5 (a) 所示，梁的长为 2700mm、宽为 450mm。两侧下端由固定刚体支撑，中间承受重为 50KG、速度为 15m/s 的刚体冲击。网格单元尺寸为 10mm，网格类型为四节点二维单元。

在经过 166 次迭代后，得到图 5-5 (b) 所示的拓扑优化结果。由图 5-5 (b) 可知，拓扑构型中承受冲击的上部区域大部分材料得以保留，下部区域留有少部分材料，中间空腔部分由两个杆状结构支撑。与图 5-5 (c) 中的对照结果^[68]进行对比，两者在总体构型上相吻合，部分细节略有差异。原因可能在于有限元模型的网

格划分和运行平台的差异。通过该优化算例，证明了混合元胞自动机法在 ABAQUS-MATLAB 拓扑优化集成框架运行的有效性。

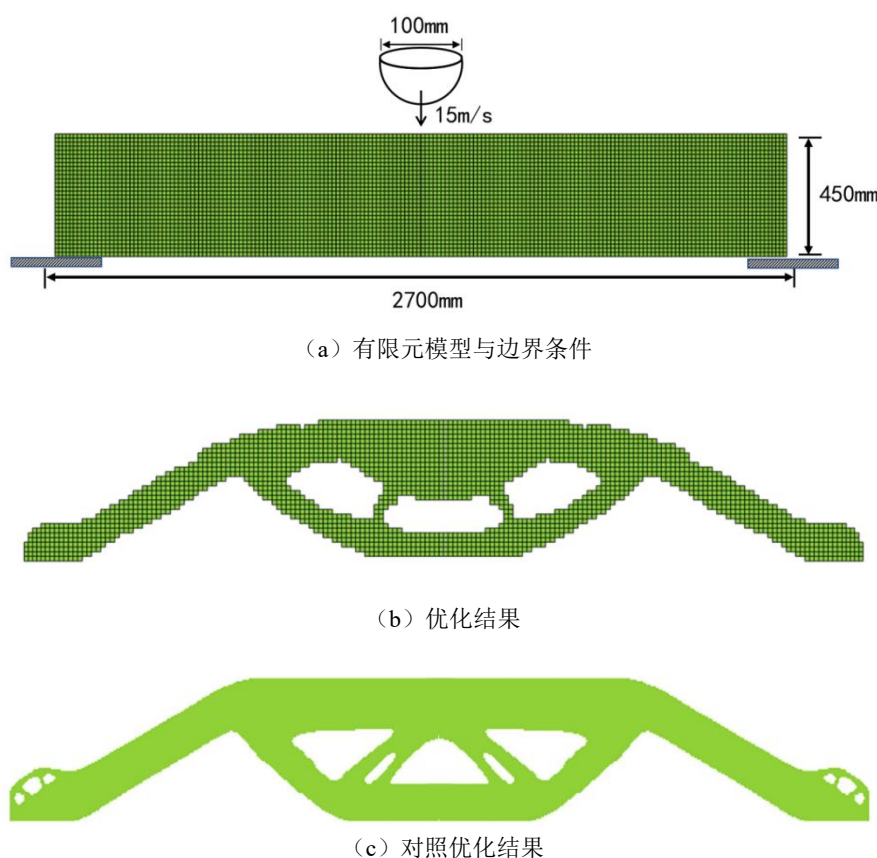


图 5-5 二维测试算例边界条件示意图与优化结果

5.4.1 不同更新规则对优化结果的影响

在 ABAQUS-MATLAB 拓扑优化集成方法中，运行时间与材料属性数有关。而设计变量的数量决定了优化模型中材料属性数。根据混合元胞自动机理论，设计变量的数量受到更新规则的控制。因此，在原比例更新规则的基础上，增加一种开-关控制规则，并对两者的运行时间和优化结果的影响进行对比。更新规则可以表示为：

$$\Delta x_i = \max\{-0.05, \min\{C_p(U_i - U^*), 0.05\}\} \quad (5.13)$$

$$\Delta x_i = C_q \times \text{sign}(U_i - U^*) \quad (5.14)$$

其中，式 (5.13) 所表示的是比例更新规则，设计变量的变化值等于该单元的内能与目标内能之间的差值与比例因子即 C_p 的乘积，且最大不会超过 0.05、最小不小于 -0.05；式 (5.14) 所示的是开-关更新规则，采用信号函数将单元的内能与目标内能之间的差值转化为信号，即 {1,0,-1}； C_q 表示开-关控制系数。对相同条件的同一设计域，分别采用式 (5.13) 和式 (5.14) 两种更新规则进行拓扑优化，比较两者

结果的异同,用以分析更新规则对拓扑优化的影响,确定适用于汽车吸能耐撞性拓扑优化的更新规则。

在 ABAQUS-MATLAB 平台采用基于 HCA 耐撞性拓扑优化集成方法,对夹紧梁分别进行不同更新规则的耐撞性拓扑优化。如图 5-6 所示,设计域为二维夹紧梁结构的一半,其全长为 800mm、宽为 200mm,该梁两端被固定,中间承受直径 50mm、重为 40KG、速度为 10m/s 的刚体冲击。

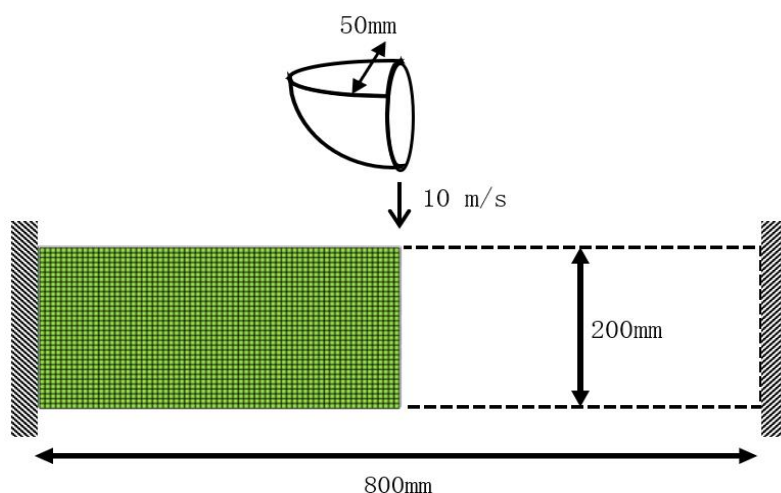
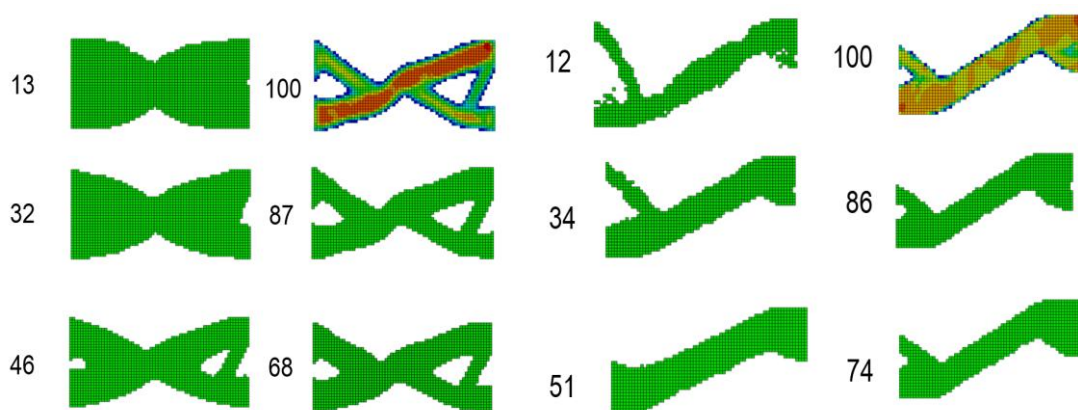


图 5-6 冲击载荷下夹紧梁的边界条件示意图

图 5-7 给出了两种更新规则下的耐撞性拓扑优化结果,其中,5-7(a)为比例更新规则的迭代过程,5-7(b)为开-关更新规则的迭代过程。可以看出,两种更新规则所得到的优化结果差异较大,比例更新规则的最终优化结果是两个杆件相互交叉的形状,而开-关型更新结果则是从承受冲击处到固定处的一体结构。虽然在构型上有所差距,但两者的应力分布存在相似之处,高应力区都是分布在顶点到固定点的斜向区域内,而开-关型跟新规则的结果较为集中,比例型更新规则的结果相对均匀。



(a) 比例更新规则

(b) 开-关更新规则

图 5-7 不同更新规则下的优化结果

两种更新规则产生不同拓扑优化构型同时，也影响了优化用时和目标内能变化。采用比例型更新规则的优化用时为 21370.1 秒，而开-关型更新规则的用时较少，只有 5126.4 秒，该结果符合两种更新规则的基本原理，即产生设计变量的个数少的更新规则用时更少。

图 5-8 为目标内能随迭代次数变化的曲线，图 5-8 (a) 为采用比例更新规则的迭代过程，图 5-8 (b) 为利用开-关更新规则的迭代过程。可以看到随着迭代次数的增加，两者目标内能的变化趋势相似，呈现逐渐上升后趋于波动的情形。但是比例更新规则的目标内能曲线变化更平稳，而开-关更新规则下的目标内能变化波动较大。该差异使得优化过程中，采用比例更新规则结构的体积分数变化比开-关更新规则更为稳定。综合上述分析，在基于混合元胞自动机法 ABAQUS-MATLAB 耐撞性拓扑优化集成方法中，与开-关型更新规则相比，采用比例型更新规则的优化过程更稳定，优化结果具有更好的细节表现，但每次迭代中会产生更多的设计变量，使得优化用时较多。

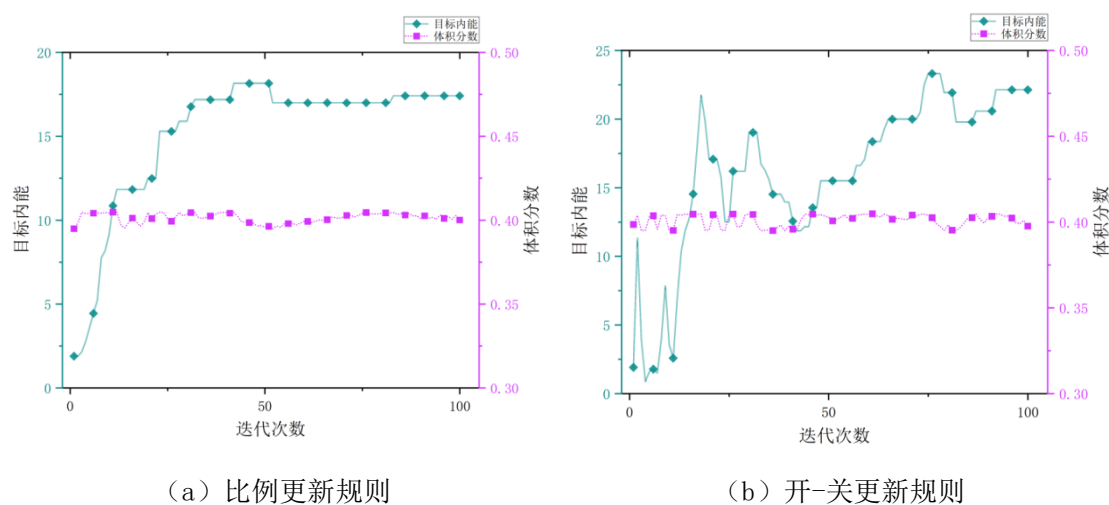


图 5-8 两种更新规则的目标内能和体积分数随迭代次数变化曲线

5.4.2 不同自动机邻域形状对优化结果的影响

为确定在三维优化问题中，诺依曼型元胞自动机和放射型元胞自动机对优化结果的影响，在 ABAQUS-MATLAB 平台采用基于 HCA 耐撞性拓扑优化集成方法，通过分别修改 Python 核心程序以使用不同的元胞自动机，对同一优化对象在相同条件下进行耐撞性拓扑优化。如图 5-9 所示，优化对象是长为 400mm、宽为 100mm 和高为 100mm 的三维夹紧梁，梁的两端固定，中间承受质量为 3KG、速度为 10m/s 的刚体冲击，优化的目标体积分数为 0.5。因该结构左右对称，为节省运算时间，采用半夹紧梁的作为设计域进行碰撞拓扑优化。

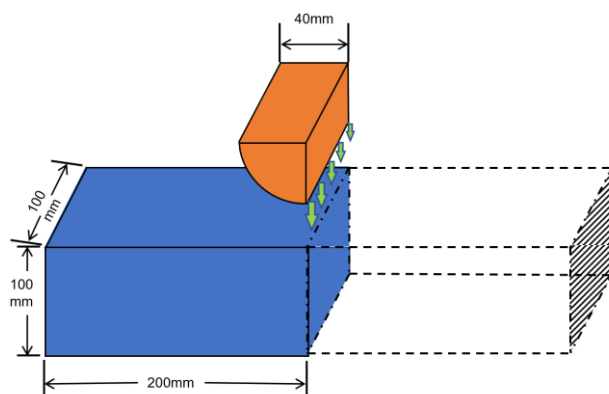
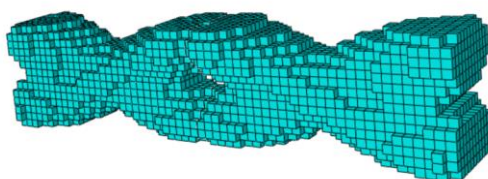
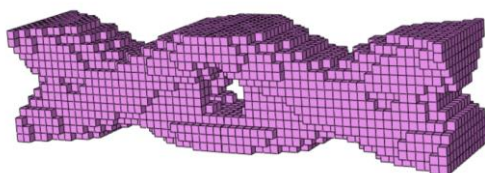


图 5-9 三维优化算例边界条件示意图

采用诺依曼型邻域的优化过程，经过 81 次迭代后得到如图 5-10 (a) 所示的优化结果，采用放射型邻域元胞自动机的优化过程，经过 60 次迭代后得到如图 5-10 (b) 所示优化结果。对比图 5-10 (a) 和图 5-10 (b) 可以看出，两者外部形状大致相同仅细节有少许差别。在优化用时方面，放射型邻域领域的运行时间比诺依曼型邻域少了近 40 分钟。



(a) 诺依曼邻域



(b) 放射型邻域

图 5-10 使用两种自动机晶格形状条件下得到的优化结果

两个优化结果在耐撞性能表现上有所差异。图 5-11 给出了采用两种领域形状优化结果的力-位移曲线。可以看到，放射型邻域的力-位移曲线与横坐标围成的区域面积大于诺依曼型邻域曲线下的相应面积，即前者的吸能效果好于后者。此外，采用放射型邻域优化结果的变形量，比诺依曼型邻域优化结果的变形量大。可见，采用放射型元胞自动机在收敛性、优化用时和吸能表现等方面，比诺依曼型元胞自动机更好。

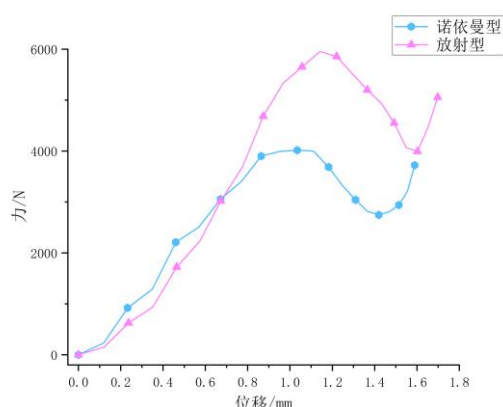


图 5-11 两个优化结果的力-位移曲线

5.4.3 吸能盒的耐撞性拓扑优化

根据前文的对比结果,综合考虑优化收敛和拓扑结果性能表现,在 ABAQUS-MATLAB 平台采用基于 HCA 耐撞性拓扑优化集成方法,采用比例更新规则和放射型邻域元胞自动机,以提升耐撞性为优化目标对汽车吸能盒进行拓扑优化。吸能盒有限元结构模型见 4.4.2 节,但需对材料属性、刚体质量和碰撞速度进行修改,优化的目标体积分数仍为 0.6。

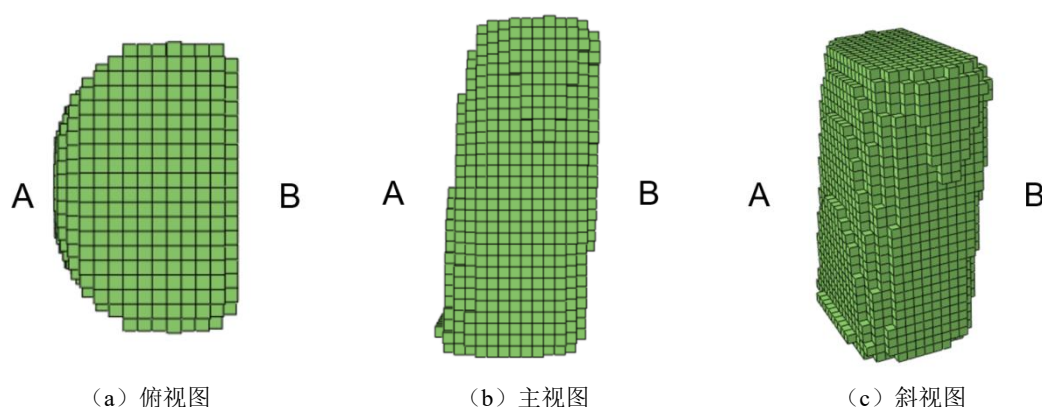


图 5-12 冲击载荷下吸能盒优化结果

图 5-12 给出了该汽车吸能盒耐撞性拓扑优化结果,其中 A 侧后承受撞击、B 侧先承受撞击。从图 5-12 可知,从顶部开始材料靠近 B 侧分布,而越接近底部材料反而向 A 侧聚集。原因在于该吸能盒受到倾斜的撞击,内部应力以 B 侧顶端到 A 侧底端呈对角线的趋势分布,故优化结果将该区域的材料保留以承受撞击。尽管采用了挤压约束条件,但由于混合元胞自动机方法的特性,以耐撞性为目标的拓扑优化结果,并没有得到如 4.4.2 节以刚性最大化为优化目标的呈现贯通的拓扑结构。

图 5-13 为吸能盒优化前后,吸能盒结构的力-位移曲线和吸能量的对比。从图 5-13 (a) 中可以看出,优化前后结构几乎在同等变形量下产生峰值撞击力,且随着变形量的增加两者受到的撞击力开始减少。但是优化后结构的撞击力在下降后

又开始上升并维持在一定范围内,而优化前的撞击力则是快速的下降。对比峰值力和吸能量的数值可知,优化后结构的降低了 34.34%,吸能量从 7130.9KJ 增长到 9949.6KJ,增长了 39.52%,结构的耐撞性得到了明显的提升。可见,采用基于 HCA 法的拓扑优化集成方法,能够在 ABAQUS-MATLAB 平台有效地对吸能盒结构进行耐撞性拓扑优化设计,为其结构设计提供指导。

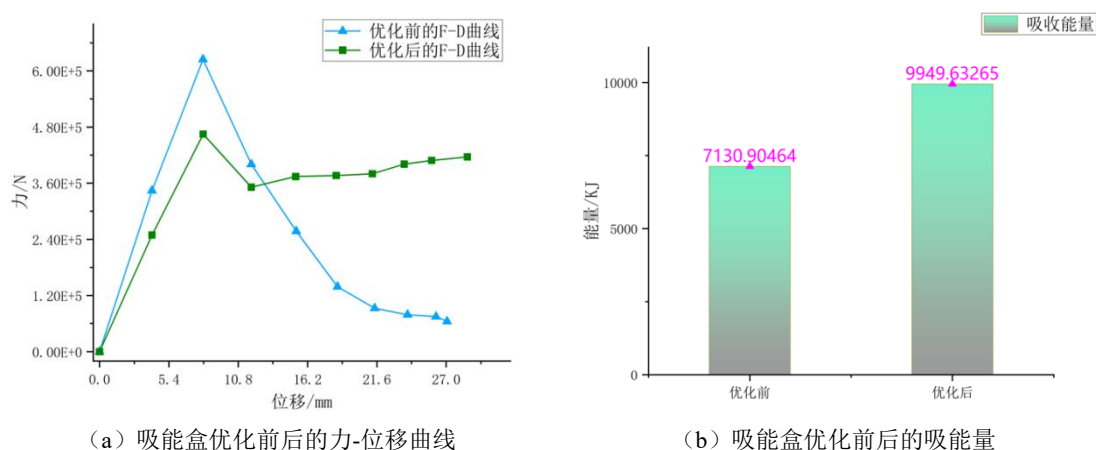


图 5-13 吸能盒优化前后的力-位移曲线与吸能量对比

5.5 本章小结

本章为提高碰撞过程中结构的耐撞性,在 ABAQUS-MATLAB 拓扑优化集成框架的基础上,按照初始化、有限元前处理、元胞自动机组建、有限元结果组装、挤压条件、单元状态更新和设计变量更新等流程,编写了适用于耐撞性拓扑优化的 MATLAB 核心程序和 Python 核心程序。构建了基于 HCA 法的 ABAQUS-MATLAB 耐撞性拓扑优化集成方法。通过与已有 HCA 动态拓扑优化结果对比,证明了该耐撞性拓扑优化集成方法能够有效地解决在 ABAQUS-MATLAB 平台中处理碰撞拓扑优化问题。

分析了不同更新规则 and 不同元胞自动机形状,对优化过程和优化结果的影响。发现采用比例更新规则,其迭代过程中目标内能变化更平稳,得到的优化结果材料分布均匀且具有更多细节,但是优化用时比开-关型更新规则要多。采用诺依曼型自动机邻域与放射型自动机邻域,所得到的优化结果外形差别较小,但是后者的优化过程迭代更快,吸能效果更好。最后,在 ABAQUS-MATLAB 平台采用基于 HCA 耐撞性拓扑优化集成方法,利用比例更新规则和放射型元胞自动机,以耐撞性为优化目标,对汽车吸能盒进行了碰撞拓扑优化研究。优化后,拓扑结构的峰值力降低了 34.34%,吸能量增长了 39.52%,耐撞性得到了明显提升。证明了本文提出的拓扑优化集成框架能够与拓扑优化方法结合进行工程应用,其结果可为产品概念设计提供参考。

第六章 总结与展望

6.1 全文总结

拓扑优化是一种在概念设计阶段用于提高产品性能、降低生产成本的结构优化方法。论文针对众多拓扑优化公开基础代码及其衍生基础代码快速工程应用需求,围绕拓扑优化集成方法、集成方法实现及其碰撞拓扑优化等方面内容,开展了基于 SIMP 法、BESO 法、ESLO 法和 HCA 法等四种方法的 ABAQUS-MATLAB 平台拓扑优化集成方法实现、拓扑优化集成方法的有效性验证及其汽车吸能盒碰撞拓扑应用研究,主要研究结论如下:

(1) 分析了 ABAQUS 软件自带脚本接口的二次开发的优势,.CAE 和.ODB 文件是生成.INP 文件和.FIL 文件的基础,相比于文本文件,采用前者即模型数据库文件进行集成,其稳定性更好、运行效率更高。针对经典拓扑优化 99 行 SIMP 法基础程序,设计了集成框架中 MATLAB 核心程序的结构,为 ABAQUS-MATLAB 拓扑优化集成方法的实现奠定了基础。提出了采用 ABAQUS 创建模型、Python 核心程序控制有限元前后处理、MATLAB 核心程序控制拓扑优化流程的 ABAQUS-MATLAB 拓扑优化集成方法。

(2) 开发了 MATLAB 核心程序和 Python 核心程序,实现了基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成方法。采用该方法对经典算例进行拓扑优化,所得结果的拓扑构型与对照结果相吻合,证明了基于 SIMP 法的 ABAQUS-MATLAB 拓扑优化集成方法的有效性。通过拓展两个核心程序,实现了在 ABAQUS-MATLAB 平台,进行大规模有限元模型、多载荷工况和装配体中部件等的拓扑优化。

(3) 实现了基于 BESO 法的 ABAQUS-MATLAB 拓扑优化集成方法,表明了 ABAQUS-MATLAB 拓扑优化集成框架的良好拓展性。对比分析了 BESO 法在 A-M 平台和 ABAQUS 单平台的运算效率,发现当过滤半径为 1.5 时,前者比后者快 2.4 倍,随过滤半径的增加差距会进一步增加,表明本文实现的拓扑优化集成方法能够提高拓扑优化方法的运行效率。

(4) 实现了基于 BESO 法和 ESLO 法的 ABAQUS-MATLAB 碰撞拓扑优化集成方法,通过与经典算例的拓扑优化结果对比,证明了碰撞拓扑优化方法在 ABAQUS-MATLAB 平台运行的有效性。在此基础上,对 Al/CFRP 混合结构吸能盒和单一材料吸能盒进行了应变能最小化的动态拓扑优化。通过对比发现,前者的最大变形量为 44.7mm,后者为 48.9mm,混合结构吸能盒的优化结果在结构稳定

性和吸能方面，均优于单材料吸能盒。分析 Al/CFRP 混合结构吸能盒在不同倾斜角度工况下的碰撞拓扑优化结果发现，小角度碰撞下 AL/CFRP 混合结构吸能盒的碰撞拓扑优化结果更为理想。

(5) 实现了基于 HCA 法的 ABAQUS-MATLAB 耐撞性拓扑优化集成方法。通过与已有 HCA 动态拓扑优化结果对比，证明了 ABAQUS-MATLAB 平台运行耐撞性拓扑优化能够有效地解决碰撞拓扑优化问题。分析了不同更新规则 and 不同元胞自动机形状对优化过程和优化结果的影响，发现采用比例更新规则，其迭代过程中目标内能变化更平稳，优化结果的材料分布均匀且具有更多细节；放射型自动机邻域比诺依曼型自动机邻域迭代更快、吸能效果更好。利用集成方法，在 ABAQUS-MATLAB 平台运行 HCA 法，进行了汽车吸能盒的碰撞拓扑优化研究。优化后，吸能盒拓扑结构的峰值力降低了 34.34%，吸能量增长了 39.52%，提升了吸能盒的耐撞性。

6.2 研究展望

本文所提出的 ABAQUS-MATLAB 拓扑优化集成框架具有较好稳定性和拓展性。在该拓扑优化集成框架下建立了针对四种不同拓扑方法的 ABAQUS-MATLAB 拓扑优化集成方法，可进行二维或三维结构、静态线性或动态非线性等复杂系统的拓扑优化设计。为众多拓扑优化方法的快速工程应用提供了一个可行的思路，通过更改优化求解器或其他子函数，即可便捷地实现拓扑优化方法的复杂问题应用。虽然采用上述拓扑优化集成方法，实现了拓扑优化基础代码的快速工程应用，但由于时间和条件的限制，存在的不足和可以继续改进的内容有：

(1) 集成方案中的两个核心程序在优化运行过程中，ABAQUS 和 MATLAB 会产生和拓扑优化不相关的中间文件，当模型单元数量或者材料属性较多时，每次迭代所产生的中间文件会占用大量的硬盘空间，后续需要进一步完善核心程序的功能，以解决该问题。

(2) 在集成方法下，采用二元设计变量的拓扑优化方法与 ABAQUS-MATLAB 平台契合度较好，具有较高的运行效率。但对于多元设计变量的拓扑优化方法，ABAQUS 在每次迭代中需创建相当数量的材料属性，导致随着随迭代步数的增加，单次迭代所需的时间也相应增加。针对该问题可从优化方法和 ABAQUS 特性两方面进一步深入研究。

参考文献

- [1] Rozvany G I N. Aims, scope, methods, history and unified terminology of computer-aided topology optimization in structural mechanics[J]. Structural and Multidisciplinary Optimization, 2001, 21(2):90-108.
- [2] Schmit L A . Structural Design by Systematic Synthesis[C]. Second Conference on Electronic Computation, 1960.
- [3] BENDSOE M P, KIKUCHI N. Generating optimal topologies in structural design using a homogenization method[J]. Computer Methods in Applied Mechanics and Engineering, 1988, 71:197-224.
- [4] Hassani B, Hinton E. Homogenization and structural topology optimization. Theory, practice and software[M]. Springer London, 1999.
- [5] BENDSOE M P, SIGMUND O. Topology Optimization: Theory, Method and Applications[M]. Springer, 2003.
- [6] Christensen P W, Klarbring A. An Introduction to Structural Optimization[J]. Solid Mechanics and its Applications, 2008, 153(1):355-376.
- [7] Rietz A. Sufficiency of a finite exponent in SIMP (power law) methods[J]. Structural and Multidisciplinary Optimization, 2001, 21(2):159-163.
- [8] Osher S J, Santosa F. Level Set Methods for Optimization Problems Involving Geometry and Constraints I. Frequencies of a Two-Density Inhomogeneous Drum[J]. Journal of Computational Physics, 2001, 171(1):272-288.
- [9] Xie Y M, Steven G P. Technical note a simple approach to Structural Optimization[J]. Computers & Structures, 1993, 49(5):885-896.
- [10] Bourdin B, Chambolle A. Design-dependent loads in topology optimization[J]. ESAIM - Control, Optimisation and Calculus of Variations, 2003,9(January): 247-273.
- [11] Zhou M, Rozvany G I N. The COC algorithm, Part II: Topological, geometrical and generalized shape optimization[J]. Computer Methods in Applied Mechanics and Engineering, 1991, 89(1-3):309-336
- [12] Eschenauer H A, Kobelev V V., Schumacher A. Bubble method for topology and shape optimization of structures[J]. Structural Optimization, 1994, 8(1):42-51
- [13] Zhang W, Li D, Yuan J, et al. A new three-dimensional topology optimization method based on moving morphable components (MMCs)[J]. Computational Mechanics, 2017, 59(4):647-665.

- [14] Zhu B, Zhang X, Zhang H, et al. Design of compliant mechanisms using continuum topology optimization: A review[J]. Mechanism and Machine Theory, 2020, 143: 103622.
- [15] Bendse M P, Sigmund O . Topology Optimization: Theory, Method and Applications[M]. Springer, 2003.
- [16] Kharmanda G, Olhoff N, Mohamed A, et al. Reliability-based topology optimization[J]. Structural and Multidisciplinary Optimization, 2004, 26(5): 295-307.
- [17] Suresh K. A 199-line Matlab code for Pareto-optimal tracing in topology optimization[J]. Structural and Multidisciplinary Optimization, 2010, 42(5):665-679.
- [18] Liu K, Tovar A. An efficient 3D topology optimization code written in Matlab[J]. Structural and Multidisciplinary Optimization, 2014, 50(6):1175-1196.
- [19] Schmidt S, Schulz V. A 2589 line topology optimization code written for the graphics card[J]. Computing and Visualization in Science, 2011, 14(6):249-256.
- [20] Andreassen E, Clausen A, Schevenels M, et al. Efficient topology optimization in MATLAB using 88 lines of code[J]. Structural and Multidisciplinary Optimization, 2011, 43(1):1-16.
- [21] Ferrari F, Sigmund O. A new generation 99 line Matlab code for compliance topology optimization and its extension to 3D[J]. Structural and Multidisciplinary Optimization, 2020, 62(4):2211-2228.
- [22] Huang X, Xie Y M. A further review of ESO type methods for topology optimization[J]. Structural and Multidisciplinary Optimization, 2010, 41(5):671-683.
- [23] Zhou S, Cadman J, Chen Y, et al. Design and fabrication of biphasic cellular materials with transport properties - A modified bidirectional evolutionary structural optimization procedure and MATLAB program[J]. International Journal of Heat and Mass Transfer, 2012, 55(25-26):8149-8162.
- [24] Da D, Xia L, Li G, et al. Evolutionary topology optimization of continuum structures with smooth boundary representation[J]. Structural and Multidisciplinary Optimization, 2018, 57(6):2143-2159.
- [25] Wang M Y, Wang X, Guo D. A level set method for structural topology optimization[J]. Computer Methods in Applied Mechanics and Engineering, 2003, 192(1-2):227-246.
- [26] Allaire G, Pantz O. Structural optimization with FreeFem++[J]. Structural and Multidisciplinary Optimization, 2006, 32(3):173-181.
- [27] Challis V J. A discrete level-set topology optimization code written in Matlab[J]. Structural and Multidisciplinary Optimization, 2010, 41(3):453-464
- [28] Laurain A. A level set-based structural optimization code using FEniCS[J]. Structural and Multidisciplinary Optimization, 2018, 58(3):1311-1334.

- [29] Zhang X, Zhu B, Topology Optimization of Compliant Mechanisms[M], Springer, 2018.
- [30] Wei P, Li Z, Li X, et al. An 88-line MATLAB code for the parameterized level set method based topology optimization using radial basis functions[J]. Structural and Multidisciplinary Optimization, 2018, 58(2):831-849.
- [31] Zhang W, Yuan J, Zhang J, et al. A new topology optimization approach based on Moving Morphable Components (MMC) and the ersatz material model[J]. Structural and Multidisciplinary Optimization, 2016, 53(6):1243-1260.
- [32] Talischi C, Paulino G H, Pereira A, et al. PolyTop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes[J]. Structural and Multidisciplinary Optimization, 2012, 45(3):329-357.
- [33] Tavakoli R, Mohseni S M. Alternating active-phase algorithm for multimaterial topology optimization problems: A 115-line MATLAB implementation[J]. Structural and Multidisciplinary Optimization, 2014, 49(4):621-642.
- [34] Biyikli E, To A C. Proportional Topology Optimization: A New Non-Sensitivity Method for Solving Stress Constrained and Minimum Compliance Problems and Its Implementation in MATLAB[J]. PLoS One, 2015, 10(12):e0145041.
- [35] Gao J, Luo Z, Xia L, et al. Concurrent topology optimization of multiscale composite structures in Matlab[J]. Structural and Multidisciplinary Optimization, 2019, 60:2621-261.
- [36] Zhu B, Zhang X, Li H, et al. An 89-line code for geometrically nonlinear topology optimization written in FreeFEM[J]. Structural and Multidisciplinary Optimization, 2021, 63(2):1015-1027.
- [37] Picelli R, Sivapuram R, Xie Y M. A 101-line MATLAB code for topology optimization using binary variables and integer programming[J]. Structural and Multidisciplinary Optimization, 2021, 63(2):935-954.
- [38] Zhuang Z, Xie Y M, Li Q, et al. A 172-line Matlab code for structural topology optimization in the body-fitted mesh[J]. Structural and Multidisciplinary Optimization, 2023, 66(1):1-18.
- [39] Han Y, Xu B, Liu Y. An efficient 137-line MATLAB code for geometrically nonlinear topology optimization using bi-directional evolutionary structural optimization method[J]. Structural and Multidisciplinary Optimization, 2021, 63(5):2571-2588.
- [40] Yaghmaei M, Ghoddosian A, Khatibi M M. A filter-based level set topology optimization method using a 62-line MATLAB code[J]. Structural and Multidisciplinary Optimization, 2020, 62(2):1001-1018.
- [41] Wang Y, Kang Z. MATLAB implementations of velocity field level set method for topology optimization: an 80-line code for 2D and a 100-line code for 3D problems[J]. Structural and Multidisciplinary Optimization, 2021, 64(6):4325-4342.

- [42] Zuo Z H, Xie Y M. A simple and compact Python code for complex 3D topology optimization[J]. *Advances in Engineering Software*, 2015, 85:1-11.
- [43] 任高晖. 基于 BESO 法的结构拓扑优化研究及应用[D]: [硕士学位论文]. 哈尔滨工程大学, 2016.
- [44] Chen Q, Zhang X, Zhu B. A 213-line topology optimization code for geometrically nonlinear structures[J]. *Structural and Multidisciplinary Optimization*, 2018, 59(5):1863-1879.
- [45] Papazafeiropoulos G, Muñiz-Calvente M, Martínez-Pañeda E. Abaqus2Matlab: A suitable tool for finite element post-processing[J]. *Advances in Engineering Software*, 2017, 105:9-16.
- [46] Antony A. Development of ABAQUS-MATLAB Interface for Design Optimization using Hybrid Cellular Automata and Comparison with Bidirectional Evolutionary Structural Optimization[D]. Purdue University Graduate School, 2022.
- [47] 王朝辉. 双向渐进结构拓扑优化在 ABAQUS-MATLAB 平台中的集成实现[D]: [硕士学位论文]. 华中科技大学, 2019.
- [48] Mayer R R, Kikuchi N, Scott R A. Application of topological optimization techniques to structural crashworthiness[J]. *Proceedings of the ASME Design Engineering Technical Conference*, 1994, Part F1678(March 1995):557-568.
- [49] Forsberg J, Nilsson L. Topology optimization in crashworthiness design[J]. *Structural and Multidisciplinary Optimization*, 2007, 33(1):1-12.
- [50] Pedersen C B W. Topology optimization design of crushed 2D-frames for desired energy absorption history[J]. *Structural and Multidisciplinary Optimization*, 2003, 25(5-6):368-382.
- [51] Torstenfelt B, Klarbring A. Conceptual optimal design of modular car product families using simultaneous size, shape and topology optimization[J]. *Finite Elements in Analysis and Design*, 2007, 43(14):1050-1061.
- [52] Ortmann C, Schumacher A. Graph and heuristic based topology optimization of crash loaded structures[J]. *Structural and Multidisciplinary Optimization*, 2013, 47(6):839-854.
- [53] Tovar A, Quevedo W I, Patel N M, et al. Hybrid cellular automata with local control rules: a new approach to topology optimization inspired by bone functional adaptation[J]. *Conference*, 2005(June):1-11.
- [54] Bandi P, Schmiedeler J P, Tovar A. Design of crashworthy structures with controlled energy absorption in the hybrid cellular automaton framework[J]. *Journal of Mechanical Design, Transactions of the ASME*, 2013, 135(9).
- [55] Afrousheh M, Marzbanrad J, Göhlich D. Topology optimization of energy absorbers under crashworthiness using modified hybrid cellular automata (MHCA) algorithm[J]. *Structural and Multidisciplinary Optimization*, 2019, 60(3):1021-1034.

- [56] Choi W S, Park G J. Structural optimization using equivalent static loads at all time intervals[J]. *Computer Methods in Applied Mechanics and Engineering*, 2002, 191(19-20):2105-2122.
- [57] Jang H H, Lee H A, Yi S IL, et al. Cross-section design of the crash box to maximize energy absorption[J]. *SAE Technical Papers*, 2011.
- [58] Lee H A, Park G J. Nonlinear dynamic response topology optimization using the equivalent static loads method[J]. *Computer Methods in Applied Mechanics and Engineering*, 2015, 283:956-970.
- [59] 夏铭,米林,万鑫铭等. 基于低速碰撞试验的吸能支架拓扑优化设计[J].*重庆理工大学学报(自然科学)*, 2012, 26(10):11-15.
- [60] Davoudi M, Kim C. Topology optimization for crashworthiness of thin-walled structures under axial crash considering nonlinear plastic buckling and locations of plastic hinges[J]. *Engineering Optimization*, 2019, 51(5):775-795.
- [61] Ren C, Min H, Ma T, et al. An effective topology optimization method for crashworthiness of thin-walled structures using the equivalent linear static loads[J]. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 2020, 234(14):3239-3255.
- [62] 石亦平,周玉蓉. ABAQUS 有限元分析实例详解[M],机械工业出版社, 2006.
- [63] 成玲,李海波.基于脚本语言的 abaqus 二次开发[J].*现代机械*, 2009, No.150(02):58-59+65.
- [64] 任春. 改进的等效静态载荷法及其在汽车结构碰撞拓扑优化中的应用研究[D]: [博士学位论文].吉林大学, 2020.
- [65] 史峰源, 李世强, 刘志芳. 冲击载荷下结构拓扑优化设计与动态响应分析[J]. *北京理工大学学报*, 2022, 42(6):10.
- [66] Guo L, Tovar A, Penninger C L, et al. Strain-based topology optimisation for crashworthiness using hybrid cellular automata[J]. *International Journal of Crashworthiness*, 2011, 16(3):239-252.
- [67] Sun G, Li S, Li G, et al. On crashing behaviors of aluminium/CFRP tubes subjected to axial and oblique loading: An experimental study[J]. *Composites Part B: Engineering*, 2018, 145(March):47-56.
- [68] Jia J, Da D, Hu J, et al. Crashworthiness design of periodic cellular structures using topology optimization[J]. *Composite Structures*, 2021, 271(April):114164.

附录 A

表 A-1 基于 SIMP 法拓扑优化平台的 MATLAB 核心程序

基于 SIMP 法拓扑优化平台的 MATLAB 程序
<pre> 1. %%%%%%%%%Calling ABAQUS for FEA topology optimization code based on SIMP method 基于 SIMP 法通 过调用 ABAQUS 进行有限元分析的 MATLAB 基础程序%%%%%%%% 2. %%%.CAE file, .Py file and this code file should be placed in the same directory. CAE 文件 Py 文件和程序 文件需要放在同一目录中%%%% 3.function topMA(maxElNum,volfrac,penal,rmin) 4.path = 'D:\test'; 5.caeName = 'Test'; 6.modelName = 'Model-test'; 7.rfid = fopen([path,'result-elenum',num2str(maxElNum),'.txt'],'w'); 8.data = strvcats(path,caeName,modelName); 9.dfid = fopen([path,'modelData.txt'],'w'); 10.[m,~] = size(data); 11.for i = 1:m 12. fprintf(dfid,'%s\n%s\n%s\n%s\n',data(i,:)); 13.end 14 fclose(dfid); 15.x = ones(maxElNum,1)*volfrac; 16.EmlN = [1:1:maxElNum]; 17.loop = 0; 18.change = 1.; 19.%%%%%%%% Start of cycle 进入循环%%%%%%%% 20.while change > 0.01 21. loop = loop + 1; 22. lfid = fopen([path,'loop.txt'],'w+'); 23. fprintf(lfid,'%d',loop); 24. fclose(lfid); 25. xold = x; 26. cd(path); 27. xlswrite(['ExM_Evo',num2str(maxElNum),'.xls'],EmlN,loop,'A1'); 28. xlswrite(['ExM_Evo',num2str(maxElNum),'.xls'],x,loop,'B1'); 29. warning off MATLAB:xlswrite:AddSheet; 30. system('abaqus cae noGUI=abaOutput.py'); 31. c = 0.; 32. dc = zeros(maxElNum,1); 33. ELSE = xlsread('ELSEandALLWK.xls',1); </pre>

```

34. c = c+ELSE(1,3)*2;
35. for em=1:maxElNum
36.     dc(em,1) = dc(em,1)-(penal*ELSE(em,2)*2)/x(em,1);
37. end
38. [dc] = check(maxElNum,rmin,x,dc);
39. [x] = OC(maxElNum,x,volfrac,dc);
40. change = max(max(abs(x-xold)));
41. disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) ...
42.     ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(maxElNum)) ...
43.     ' ch.: ' sprintf('%6.3f',change )])
44. fprintf(rfid,[' \n It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) ...
45.     ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(maxElNum)) ...
46.     ' ch.: ' sprintf('%6.3f',change )]);
47.end
48.fclose(rfid);
49. %%%%%%%%%% OPTIMALITY CRITERIA UPDATE 设计变量更新子程序%%%%%%%%%%
50.function [xnew] = OC(maxElNum,x,volfrac,dc)
51.l1 = 0; l2 = 100000; move = 0.2;
52.while (l2-l1 > 1e-4)
53.    lmid = 0.5*(l2+l1);
54.    xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid))));
55.    if sum(sum(xnew)) - volfrac*maxElNum > 0
56.        l1 = lmid;
57.    else
58.        l2 = lmid;
59.    end
60.end
61.end
62.%%%%%%%%%%%%% MESH-INDEPENDENCY FILTER 网格过滤子程序%%%%%%%%%%%%%
63.function[dcn] = check(maxElNum,rmin,x,dc)
64.oData = xlsread('EL_coordinateS.xls');
65.vMax = max(oData);
66.vMin = min(oData);
67.if vMax(4) == vMin(4)
68.    p = 3;
69.else
70.    p = 4;
71.end
72.numArrays = p-1;
73.S = cell(numArrays,1);
74.dcn = zeros(maxElNum,1);
75.for i = 1:maxElNum

```

```

76.    suma = 0;
77.    major = oData(i,:);
78.    for k = 2:p
79.        [rowa,~] = find(oData(:,k)>oData(i,k)-(rmin+1) & oData(:,k)<oData(i,k)+(rmin+1));
80.        S{k-1} = [rowa];
81.    end
82.    if p == 3
83.        eleO = intersect(S{1},S{2});
84.    else
85.        eleO = intersect(intersect(S{1},S{2}),S{3});
86.    end
87.    for j = 1:length(eleO)
88.        minor = oData(eleO(j),:);
89.        dis = sqrt((major(2)-minor(2))^2+(major(3)-minor(3))^2+(major(4)-minor(4))^2);
90.        if rmin-dis>0
91.            fac = rmin-dis;
92.            suma = suma+max(0,fac);
93.            dcn(i,1) = dcn(i,1)+max(0,fac)*x(eleO(j),1)*dc(eleO(j),1);
94.        end
95.    end
96.    dcn(i,1) = dcn(i,1)/(x(i,1)*suma);
97.end
98.end
99.end

```


附录 B

表 B-1 基于 SIMP 法拓扑优化平台的 Python 核心程序

基于 SIMP 法拓扑优化平台的 Python 程序
<pre> 1.from odbAccess import openOdb 2.import xlrd 3.import os 4.import xlwt 5.import step 6.fPath = os.getcwd()+r'\modelData.txt' 7.with open(fPath,'r') as Mas: 8. data = Mas.readlines() 9.path,caeName,modelName = data[0].strip('\n').replace(' ',''),data[1].strip('\n').replace(' ',''),data[2].strip('\n').replace(' ','') 10.with open(path+'loop.txt','r') as f: 11. data = f.readlines() 12. loop = int(data[0])-1 13.#####Gets the design variable 设计变量获取子函数##### 14.def get_data(dirCase,sheetNum): 15. data = xlrd.open_workbook(dirCase) 16. table = data.sheets()[sheetNum] 17. nor = table.nrows 18. nol = table.ncols 19. dictO = {} 20. for i in range(0,nor): 21. for j in range(nol): 22. eNum = table.cell_value(i,0) 23. xDen = table.cell_value(i,1) 24. dictO[eNum] = xDen 25. yield dictO 26.#####main function 主函数##### 27.if __name__ == '__main__': 28. myMdb = openMdb(pathName=caeName+'.cae') 29. mdl = myMdb.models[modelName] 30. part = mdl.parts['Part-1'] 31. ems,nds=part.elements,part.nodes 32. dict1={} 33. dict2={} 34. for i in get_data(path+'ExM_Evo'+str(ems[-1].label)+'.xls',loop): 35. dict1.update(i) </pre>

```

36.     for key,value in dict1.items():
37.         if dict2.__contains__(value):
38.             dict2[value].append(key)
39.         else:
40.             dict2[value]=[key]
41.     penal=3
42.     youngOld=1
43.     indexMaterial=0
44.     listDx=list(dict2.keys())
45.     for i in range(0,len(dict2)):
46.         j=listDx[i]
47.         young=j**penal*youngOld
48.         indexMaterial+=1
49.         l=str(indexMaterial)
50.         ut=list(map(int,dict2[j]))
51.         mdl.Material('Material'+l).Elastic(((young,0.3),))
52.         mdl.HomogeneousSolidSection('Sec'+l,'Material'+l)
53.         part.SectionAssignment(part.SetFromElementLabels (name='Eset-
'+modelName+l,elementLabels=ut,unsorted=True),sectionName='Sec'+l)
54.         mdl.FieldOutputRequest('SEDensity','Step-1',variables=('ELSE',))
55.         mdl.HistoryOutputRequest('ExtWork','Step-1',variables=('ALLWK',))
56.         myJob=myMdb.Job('Design'+modelName+'_job'+str(loop),modelName)
57.         myJob.submit()
58.         myJob.waitForCompletion()
59.         odb=openOdb('Design'+modelName+'_job'+str(loop)+'.odb')
60.         elseV=odb.steps['Step-1'].frames[-1].fieldOutputs['ELSE'].values
61.         obj=odb.steps['Step-1'].historyRegions ['Assembly ASSEMBLY'].historyOutputs['ALLWK'].data[-1][1]
62.         wb = xlwt.Workbook(encoding='utf-8')
63.         rSheet= wb.add_sheet('Sheet1')
64.         rawDc={}
65.         for en in elseV:rawDc[en.elementLabel]=en.data
66.         num=list(rawDc.keys())
67.         val=list(rawDc.values())
68.         for indexy in ems:
69.             indexny=indexy.label-1
70.             rSheet.write(indexny,0,num[indexny])
71.             rSheet.write(indexny,1,val[indexny])
72.         rSheet.write(0,2,obj)
73.         wb.save('ELSEandALLWK.xls')
74.         odb.close()
75.         #####Output element coordinates 单元中心坐标输出部分#####
76.         if loop==0:

```



```
77.     workbook = xlwt.Workbook(encoding='utf-8')
78.     sheet = workbook.add_sheet('EM_coordinates')
79.     c0={}
80.     for el in ems:
81.         ndc=el.connectivity
82.         c0[el.label]=[sum([nds[nd].coordinates[i]/len(ndc)for nd in ndc]) for i in range(3)]
83.         eu=int(el.label)-1
84.         sheet.write(eu,0,el.label)
85.         sheet.write(eu,1,c0[el.label][0])
86.         sheet.write(eu,2,c0[el.label][1])
87.         sheet.write(eu,3,c0[el.label][2])
88.     workbook.save('EL_coordinateS.xls')
```


附录 C

表 C-1 基于 SIMP 法拓扑优化平台的 Python 拓展程序 1

基于 SIMP 法拓扑优化平台的 Python 拓展程序 1
<pre> 1.import numpy as np#使用. mat 作为媒介的 Python 程序拓展部分 2.import scipy.io as scio 3.def get_data(p,n): 4. dataFile = p+r'\xValue'+str(loop+1)+''.mat' 5. xLoad = scio.loadmat(dataFile) 6. xValue = xLoad['x'].tolist() 7. dictOri = {} 8. dictFin = {} 9. for i in range(0,n): 10. indexL_a = int((i+1)/30000.1) 11. indexR_a = i-indexL_a*30000 12. dictOri[i+1] = xValue[indexR_a][indexL_a] 13. for key,value in dictOri.items(): 14. if dictFin.__contains__(value): 15. dictFin[value].append(key) 16. else: 17. dictFin[value]=[key] 18. return dictFin </pre>

表 C-2 基于 SIMP 法拓扑优化平台的 MATLAB 拓展程序 1

基于 SIMP 法拓扑优化平台的 MATLAB 拓展程序 1
<pre> 1. %%%%%%%%%Mesh Filtering Prepare 网格过滤的准备工作%%%%%%%% 2.function[elCage,o] = prepro(path,maxElNum,dc,rmin) 3.xV = struct2cell(load([path,'\elCoor0.mat'],'elCoorValue0')); 4.yV = struct2cell(load([path,'\elCoor1.mat'],'elCoorValue1')); 5.zV = struct2cell(load([path,'\elCoor2.mat'],'elCoorValue2')); 6.o = [xV;yV;zV]; 7.if max((o{3})) == min((o{3})) 8. p=2; 9.else 10. p=3; 11.end 12.S = cell(p,1); 13.elCage = cell(size(dc)); 14.for indexO = 1: maxElNum 15. indexR_a = indexO-floor(indexO/30000.1)*30000; 16. indexL_a = floor(indexO/30000.1)+1; </pre>

```

17.   for k=1:3
18.       [t,r]=find(o{k}>o{k}(indexR_a,indexL_a) - rmin & o{k}<o{k}(indexR_a,indexL_a) + rmin);
19.       vv = (r-1)*30000+t;
20.       S{k} = vv;
21.   end
22.   elCage{indexR_a,indexL_a}=intersect(intersect(S{1},S{2}),S{3});
23.end
24.end
25. %%%%%%%%% MESH-INDEPENDENCY FILTER 扩展后的网格过滤功能%%%%%%%%
26.function[dcn]=check(maxElNum,rmin,x,dc,path)
27.dcn = zeros(size(dc));
28.oriElCoor = struct2cell(load([path,'\oriCoor.mat'],'o'));
29.oriElCoorV = oriElCoor{1};
30.elCageM = struct2cell(load([path,'\elCage.mat'],'elCage'));
31.elCageMes = elCageM{1};
32.for indexF = 1: maxElNum
33.   indexR_b = indexF-floor(indexF/30000.1)*30000;
34.   indexL_b = floor(indexF/30000.1)+1;
35.   major = [oriElCoorV{1}(indexR_b,indexL_b),oriElCoorV{2}(indexR_b,indexL_b),oriElCoorV{3}(indexR_b,indexL_b)];
36.   suma = 0;
37.   eleO = elCageMes{indexR_b,indexL_b};
38.   for ind = 1:length(eleO)
39.       indexR_c = eleO(ind)-floor(eleO(ind)/30000.1)*30000;
40.       indexL_c = floor(eleO(ind)/30000.1)+1;
41.       minor = [oriElCoorV{1}(indexR_c,indexL_c),oriElCoorV{2}(indexR_c,indexL_c),oriElCoorV{3}(indexR_c,indexL_c)];
42.       dis = sqrt((major(1)-minor(1))^2+(major(2)-minor(2))^2+(major(3)-minor(3))^2);
43.       if rmin-dis>0
44.           fac = rmin-dis;
45.           suma = suma+max(0,fac);
46.           dcn(indexR_b,indexL_b)=dcn(indexR_b,indexL_b)+max(0,fac)*x(indexR_c,indexL_c)*dc(indexR_c,indexL_c);
47.       end
48.   end
49.   dcn(indexR_b,indexL_b) = dcn(indexR_b,indexL_b)./(x(indexR_b,indexL_b)*suma);
50.end
51.dcn(dcn == 0) = nan;
52.end

```

表 C-3 基于 SIMP 法拓扑优化平台的 MATLAB 拓展程序 2

基于 SIMP 法拓扑优化平台的 MATLAB 拓展程序 2

```

7. loadMax='2';%多载荷工况下 MATLAB 程序的拓展及替换内容
9. data=strvcat(data,path,caeName,modelName,loadMax);

34.for m=1:str2num(loadMax)
35.    ELSE = xlsread('ELSEandALLWK.xls.xls',m);
36.    c = c + ELSE(1,3)*2;
37.    for em=1:MaxCellNum
38.        dc(em,1) = dc(em,1)-(penal*ELSE(em,2)*2)/x(em,1);
39.    end
40. end

```

表 C-4 基于 SIMP 法拓扑优化平台的 Python 拓展程序 2

基于 SIMP 法拓扑优化平台的 Python 拓展程序 2

```

11.path,caeName,modelName,load=data[0].strip('\n').replace(' ',''),data[1].strip('\n').replace(' ',''),data[2].strip('\n').replace(' ',''),data[3].strip('\n').replace(' ','')#多载荷工况下针对 Python 基础程序的拓展内容
12.loadMax=int(load)+1

61.wb = xlwt.Workbook(encoding='utf-8')
62.    dictT={}
63.    for indexM in range(1,loadMax):
64.        seng=odb.steps['Step-'+str(indexM)].frames[-1].fieldOutputs['ELSE'].values
65.        obj=odb.steps['Step-'+str(indexM)].historyRegions['Assembly
ASSEMBLY'].historyOutputs['ALLWK'].data[-1][1]
66.        dictT['ns{0}'.format(indexM)]=ens'+str(indexM)
67.        dictT['ns{0}'.format(indexM)] = wb.add_sheet('Sheet'+str(indexM))
68.        rawDc={}
69.        for en in seng:rawDc[en.elementLabel]=en.data
70.        num=list(rawDc.keys())
71.        val=list(rawDc.values())
72.        for indexy in Elmts:
73.            indexny=indexy.label-1
74.            dictT['ns{0}'.format(indexM)].write(indexny,0,num[indexny])
75.            dictT['ns{0}'.format(indexM)].write(indexny,1,val[indexny])
76.            dictT['ns{0}'.format(indexM)].write(0,2,obj)
77.    wb.save('ELSEandALLWK.xls')

```

表 C-5 基于 SIMP 法拓扑优化平台的 Python 拓展程序 3

基于 SIMP 法拓扑优化平台的 Python 拓展程序 3

```

6. import regionToolset#针对装配结构对原 Python 的拓展及修改部分

33.assembly = mdl.rootAssembly

```

```
34.Elpl = assembly.instances['Part-1-1'].elements
35.myRegion = regionToolset.Region(elements = Elpl)
36.assembly.Set(name='mySet',region=myRegion)
37.targetRegion = assembly.sets['mySet']
38.assembly = mdl.rootAssembly

61. mdl.FieldOutputRequest('SEDensity','Step-1', variables=('ELSE',),region = targetRegion)
```

附录 D

表 D-1 基于混合元胞自动机方法的 MATLAB 核心程序

基于混合元胞自动机方法集成平台的 MATLAB 程序
<pre> 1.function maHCA(maxElNum,tMass) 2.%% initialization 初始化阶段 3.path = 'D:\SimpleHca'; 4.caeName='crashbox'; 5.modelName='Model-1'; 6.cd(path); 7.data=char(path,caeName,modelName); 8.dfid=fopen([path,'\modelData.txt'],'w'); 9.[m,~]=size(data); 10.for i =1:m 11. fprintf(dfid,'%s\n%s\n%s\n%s\n',data(i,:)); 12.end 13.fclose(dfid); 14.x = ones(maxElNum,1)*(tMass/maxElNum); 15.loop = 0; 16.xPro = zeros(size(x)); 17.iedPro = zeros(size(x)); 18.change = inf; 19.loopMax = 150; 20.%% Circulation 进入循环 21.while (change>maxElNum*0.001 && loop<loopMax) 22. loop = loop+1; 23. lfid = fopen([path,'\loop.txt'],'w+'); 24. fprintf(lfid,'%d',loop); 25. fclose(lfid); 26. save('xValue.mat','x'); 27. system('abaqus cae noGUI=SrEaba_out.py'); 28. eledRaw = cell2mat(struct2cell(load([path,'\SED.mat'],'esedenV'))); 29. pedRaw = cell2mat(struct2cell(load([path,'\PED.mat'],'epddenV'))); 30. eaRaw = cell2mat(struct2cell(load([path,'\EASE.mat'],'easedenV'))); 31. cA = cell2mat(struct2cell(load([path,'\cA.mat'],'caValue'))); 32. iedRaw = eledRaw+pedRaw+eaRaw; 33. xfCa = [x;0]; 34. iedfCa = [iedRaw;0]; 35. if loop ==1 36. tIED = (sum(sum(iedRaw))/maxElNum)*0.4; </pre>

```

37. end
38. xFc = x;
39. %% EXTRUSION 挤压条件
40. if loop == 1
41.     dcNewO = newO;
42.     save('dirZ.mat','dcNewO');
43. end
44. dcOm = struct2cell(load([path,'\dirZ.mat'],'dcNewO'));
45. dcOr = dcOm{1,1};
46. [~,ml] = size(dcOr);
47. for sInd = 1:ml
48.     iedRaw(dcOr(:,sInd),1) = sum(iedRaw(dcOr(:,sInd),1))/size(dcOr(:,sInd),2);
49. end
50. %% local approach 局部策略
51. for xIn = 1:maxElNum
52.     xPro(xIn,1) = sum(xfCa(cA(xIn,:)),1)/size(cA(1,:),2);
53. end
54. for iedIn = 1:maxElNum
55.     iedPro(iedIn,1) = sum(iedfCa(cA(iedIn,:)),1)/size(cA(1,:),2);
56. end
57. iedProD = iedPro;
58. if loop > 1
59.     iedProD = (iedPro+iedProOld)/2;
60. end
61. iedProOld = iedPro;
62. xUp = zeros(size(xPro));
63. %% UPDATE 设计变量更新阶段
64. massFc = sum(sum(x));
65. [j,~] = find(xPro>0.001 & xPro<1.000);
66. xUp(j,1) = 0.25*sign(iedProD(j,1)-tIED);
67. xPro(j,1) = xPro(j,1) + xUp(j,1);
68. x = xPro;
69. x(x<0.001) = 0.001;
70. x(x>1.000) = 1.000;
71. mass = sum(sum(x));
72. while (abs(mass-tMass) > maxElNum*0.005)%设计变量更新内循环
73.     for xIn = 1:maxElNum
74.         xPro(xIn,1) = sum(xfCa(cA(xIn,:)),1)/size(cA(1,:),2);
75.     end
76.     xUp = zeros(size(xPro));
77.     [j,~] = find(xPro>0.001 & xPro<1.000);
78.     xUp(j,1) = 0.25*sign(iedProD(j,1)-tIED);

```



```

79.      xPro(j,1) = xPro(j,1) + xUp(j,1);
80.      x = xPro;
81.      x(x<0.001) = 0.001;
82.      x(x>1.000) = 1.000;
83.      mass = sum(sum(x));
84.      tIED=tIED*(mass/tMass);
85.  end
86.  xC = sum(abs(x-xFc));
87.  SE = sum(sum(eledRaw));
88.  mC = abs(mass - massFc);
89.  if loop >= 2
90.      change = xC+xCold;
91.  end
92.  xCold = xC;
93.  mCOld = mC;
94.  %% Output
95.  fprintf('loop.: %3i SE.:%8.4f Mtol.:%6.4f U*:%6.4f change.:%3.4f,loop,SE,mass,tIED,change);
96.end
97.function re = newO()
98.oData=xlsread('EL_coordinateS.xls');
99.oData = single(oData);
100.zd = oData(1,4);
101.[zr,~] = find(oData(:,4) == zd);
102.M = oData(zr,:);
103.[r,~] = size(M);
104.re=[];
105.for i = 1:r
106.    [fr,~] = find(oData(:,2)==M(i,2) & oData(:,3)==M(i,3));
107.    re = [re fr];
108.end
109.end
110.end

```


附录 E

表 E-1 基于混合元胞自动机方法的 Python 核心程序

基于混合元胞自动机方法集成平台的 Python 程序

```

1. if __name__ == '__main__': #HCA 法 Python 程序的主函数
2.     myMdb = openMdb(pathName=caeName+'.cae')
3.     mdl = myMdb.models[modelName]
4.     part = mdl.parts['Part-1']
5.     Elmts, Nds = part.elements, part.nodes
6.     assembly = mdl.rootAssembly
7.     eleee = assembly.instances['Part-1-1'].elements
8.     nodeee = assembly.instances['Part-1-1'].nodes
9.     myRegion = regionToolset.Region(elements = eleee)
10.    assembly.Set(name='mySet', region=myRegion)
11.    targetRegion = assembly.sets['mySet']
12.    xCoE = get_data(path, Elmts[-1].label)
13.    penal=1 #材料属性更新部分
14.    youngOld=70e3
15.    indexMaterial=0
16.    listDx=list(xCoE.keys())
17.    for i in range(0, len(xCoE)):
18.        j=listDx[i]
19.        penalXe = j**penal
20.        jl = j
21.        penalXp = jl**penal
22.        saK = [0]
23.        svK = [180.0, 190.0, 197.0, 211.5, 225.8, 233.6, 238.5, 248.5]
24.        hK = [1000.0, 700.0, 483.3, 286.0, 156.0, 98.0, 50.0]
25.        stK = [x*penalXp for x in svK]
26.        hK = [y*penalXp for y in hK]
27.        indexMaterial+=1
28.        l=str(indexMaterial)
29.        ut=list(map(int, xCoE[j]))
30.        for i in range(1, 8):
31.            sa = (stK[i]+saK[i-1]*hK[i-1]-stK[i-1])/hK[i-1]
32.            saK.append(sa)
33.        plasticc=zip(stK, saK)
34.        mdl.Material(name='Material'+l)
35.        mdl.materials['Material'+l].Elastic(table=((penalXe*youngOld, 0.33), ))
36.        mdl.materials['Material'+l].Density(table=((2.7e-09*penalXe, ), ))
37.        dMar = []

```

```

38.         for ei in range(1,8):
39.             vfP = plasticc[ei][1]-plasticc[ei-1][1]
40.             if vfP < 1e-4:
41.                 dMar.append(ei)
42.             dMar.reverse()
43.             for ite in dMar:
44.                 plasticc.pop(ite)
45.             plasticc = tuple(plasticc)
46.             mdl.materials['Material'+l].Plastic(plasticc)
47.             mdl.HomogeneousSolidSection('Sec'+l,'Material'+l)
48.             part.SectionAssignment(part.SetFromElementLabels(name='Eset-'+modelName+l,elementLabels =
ut,unsorted=True),sectionName='Sec'+l)
49.             pass
50.             myJob=myMdb.Job(caeName+'_job'+modelName+str(loop+1),modelName)#创建有限元分析作业
51.             myJob.submit()
52.             myJob.waitForCompletion()
53.             odb = openOdb(caeName+'_job'+modelName+str(loop+1)+'.odb')#有限元后处理部分
54.             sed = odb.steps['Step-1'].frames[-1].fieldOutputs['ELSE'].values
55.             ped = odb.steps['Step-1'].frames[-1].fieldOutputs['ELPD'].values
56.             ea = odb.steps['Step-1'].frames[-1].fieldOutputs['ELASE'].values
57.             sedR_array = np.zeros((Elmts[-1].label,1))
58.             pedR_array = np.zeros((Elmts[-1].label,1))
59.             eaR_array = np.zeros((Elmts[-1].label,1))
60.             for sedI in sed:
61.                 sedR_array[sedI.elementLabel-1,0] = sedI.data
62.             scio.savemat("SED",{ "esedenV":sedR_array})
63.             for pedI in ped:
64.                 pedR_array[pedI.elementLabel-1,0] = pedI.data
65.             scio.savemat('PED',{ 'epddenV':pedR_array})
66.             for eaI in ea:
67.                 eaR_array[eaI.elementLabel-1,0] = eaI.data
68.             scio.savemat('EASE',{ 'easedenV':eaR_array})
69.             odb.close()
70.             if loop == 0:#元胞自动机创建部分
71.                 elNuNd = []
72.                 ndNuEl = []
73.                 for el in eleec:
74.                     Nvav = []
75.                     ndN = el.getNodes()
76.                     for i in range(0,len(ndN)):
77.                         Nvav.append(ndN[i].label)
78.                     elNuNd.append(Nvav)

```

```

79.     for nd in nodee:
80.         Evav = []
81.         elN = nd.getElements()
82.         for j in range(0,len(elN)):
83.             Evav.append(elN[j].label)
84.         ndNuEl.append(Evav)
85.     finalV = []
86.     dl = 0
87.     if str(eleee[-1].type) == 'CPS4R':
88.         m = list(combinations(range(0,4),2))
89.         for ite in range(len(elNuNd)):
90.             mid = []
91.             dl += 1
92.             for i in range(len(elNuNd[ite])):
93.                 mid.append(ndNuEl[elNuNd[ite][i]-1])
94.             finalV.append(mid)
95.         result = []
96.         for itm in range(len(finalV)):
97.             qe = []
98.             for k in range(len(m)):
99.                 fn = list(set(finalV[itm][m[k][0]].intersection(set(finalV[itm][m[k][1]])))
100.                 qe.extend(fn)
101.                 result.append(list(set(qe)))
102.             cAV = np.zeros((Elmts[-1].label,5))
103.             for itte in range(len(result)):
104.                 result[itte].extend([Elmts[-1].label+1]*(5-len(result[itte])))
105.                 for ittz in range(len(result[itte])):
106.                     cAV[itte,ittz] = result[itte][ittz]
107.             scio.savemat('cA',{'caValue':cAV})
108.     if str(eleee[-1].type) == 'C3D8R':
109.         m = list(combinations(range(0,8),4))
110.         for ite in range(len(elNuNd)):
111.             mid = []
112.             dl += 1
113.             for i in range(len(elNuNd[ite])):
114.                 mid.append(ndNuEl[elNuNd[ite][i]-1])
115.             finalV.append(mid)
116.         result = []
117.         for itm in range(len(finalV)):
118.             qe = []
119.             for k in range(len(m)):
120.                 fn = list(set(finalV[itm][m[k][0]].intersection(set(finalV[itm][m[k][1]]),

```

```
set(finalV[itm][m[k][2]],set(finalV[itm][m[k][3]]))
121.         qe.extend(fn)
122.         result.append(list(set(qe)))
123.         cAV = np.zeros((Elmts[-1].label,7))
124.         for itte in range(len(result)):
125.             result[ite].extend([Elmts[-1].label+1]*(7-len(result[ite])))
126.             for itt in range(len(result[ite])):
127.                 cAV[ite,itt] = result[ite][itt]
128.         scio.savemat('cA',{'caValue':cAV})
129.         workbook = xlwt.Workbook(encoding='utf-8')
130.         sheet = workbook.add_sheet('EM_coordinates')
131.         c0={}
```