深 圳 大 学

本 科 毕 业 论 文（设计）

题目: 基于 FreeCAD 和 SAP2000 的平面管桁架参数化 BIM 建模及结构设计程序开发

姓名: 林凯杰

专业: 土木工程（创新班）

学院: 土木与交通工程学院

学号: 2021090102

指导教师: 陈贤川

职称: 副教授

2025 年 04 月 17 日

深圳大学本科毕业论文（设计）诚信声明

本人郑重声明：所呈交的毕业论文（设计），题目《基于 FreeCAD 和 SAP2000 的平面管桁架参数化 BIM 建模及结构设计程序开发》 是本人在指导教师的指导下，独立进行研究工作所取得的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明。除此之外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。本人完全意识到本声明的法律结果。

毕业论文（设计）作者签名： 

日期：　　2025 年 4 月 17 日

目录

基于 FreeCAD 和 SAP2000 的平面管桁架参数化 BIM 建模及结构设计程序开发

深圳大学土木与交通工程学院

姓名：林凯杰　　学号：2021090102

指导教师：陈贤川

【摘要】:针对当前 BIM 技术在钢结构正向设计中存在的软件交互壁垒与智能化不足问题，本研究以平面管桁架为对象，开发了基于 FreeCAD 平台的参数化 BIM 协同设计系统。通过构建"管道-过滤器"架构的三级程序模块，创新性地实现了管桁架的“几何建模-荷载识别-结构验算”的全流程集成。基于 FreeCAD 开发参数化建模程序，建立平面管桁架的三维 BIM 模型，并与 SAP2000 实时同步；集成大语言模型（DeepSeek）开发智能荷载工具，实现自然语言指令向结构化荷载参数的转换，并以三维箭头符号实现荷载的可视化；建立符合 GB50017-2017 规范的三级验算体系，使用 COM 接口实现 SAP2000 有限元模型的内力数据回传与构件自动校核。研究成果为钢结构智能设计提供了开源解决方案，推动了 BIM 技术在设计阶段的深度应用。

【关键词】: 建筑信息模型；参数化设计；自然语言处理；钢结构桁架；FreeCAD

1 绪论

1.1 研究背景

随着人工智能、大数据、物联网等信息技术的发展，建筑信息化在建筑工业化进程中的重要性日益突显[1]。在“ 十四五” 规划中，提出的主要任务之一为加快智能建造与新型建筑工业化协同发展，其中强调了建筑信息模型（BIM）技术的集成应用以及装配式建筑设计生产体系的发展构建[2]。为实现工程建设领域向工业化、数字化、现代化的转型升级，需要充分发挥 BIM 技术以及装配式建造各自的技术优势，实现二者的深度融合[3]。BIM 贯穿于建筑工业化的全过程[4]，对建筑工业化的发展和工程建造的转型升级具有极其重要的作用。

基于 BIM 技术，装配式建筑可实现可视化设计、数字化生产、精细化施工、信息化运维和协同化管理，从而在整个建设周期的视角上去解决建造过程中所遇到的工程问题[1]。但 BIM 技术在我国建筑行业的应用过程中，存在 BIM 人才缺失和 BIM 软件交互性差两项主要问题[5]。现有的 BIM 软件生态尚不成熟，由于国内外设计规范不同，国外 BIM 设计软件在我国本土化落地需要大量的二次开发，同时 BIM 软件功能单一且存在转换壁垒，不同软件间的衔接性差，这也增加了 BIM 的使用成本[6]。BIM 应用所存在的困难导致 BIM 技术并未真正深入建筑结构的设计、施工管理阶段，由此，开发一套具有普适性的基于 BIM 技术的结构设计方法具有重要意义[7]。

钢结构作为建筑工业化进程中优势最为突出的结构类型，因其高强度、轻质、施工速度快以及可回收利用等特点，成为现代建筑工业化发展的核心方向之一。而桁架结构作为钢结构中最为常用的结构体系之一，凭借其独特的力学性能和构造特点，在建筑领域得到了广泛应用。桁架结构通过将杆件以三角形单元组合形成稳定的受力体系，不仅能够高效传递荷载，还具有结构轻巧、用材经济、施工便利等显著特点，这些特点赋予了其易于建造大跨度结构的优势[8]。此外，桁架结构的布置形式灵活多样，能够适应不同的建筑功能和美学需求，其应用范围涵盖工业厂房、桥梁、塔架以及现代建筑的空间结构设计等领域。然而，正是由于其布置灵活、形式多样，桁架结构的设计过程往往较为复杂，尤其是在面对大跨度、异形或荷载条件特殊的情况时，传统设计方法难以高效应对。

本人在本科阶段进行钢结构课程设计时，发现钢桁架传统设计方法存在着建模繁琐、有限元模型可视性不强、构件截面修改流程繁琐等问题，这些问题的存在极大的降低了设计效率，将设计师困在机械劳动之中，对设计人员的身心健康损害十分巨大。

BIM 技术的引入为桁架结构设计提供了全新的解决方案，可以显著提高桁架设计的效率。通过桁架结构的 BIM 参数化建模，面对不同尺寸的建筑，只需修改参数即可实现模型的动态调整。BIM 模型集成了结构的几何、截面和材料等设计信息，将含有设计信息的 BIM 模型导入有限元计算软件当中，即可快速完成结构的内力计算和性能评估，从而实现从设计到分析的无缝衔接。

为将 BIM 技术深度融入建筑工业化进程，实现参数化设计的数字化与智能化目标，仍需克服诸多技术困难。开发一套高效、可视化的结构设计方法具有重要理论与实践价值。以钢桁架结构为切入点是一个代表性选择，其设计过程涉及复杂的几何建模与力学分析，能够充分体现 BIM 技术的优势。通过构建基于 BIM 的钢桁架设计方法，可为类似结构体系提供参考，推动建筑行业向数字化与智能化转型。

1.2 国内外研究现状

1.2.1 钢结构设计的发展现状

钢结构设计技术体系正在经历从传统经验设计向数字化设计的重要转型，其转型主要体现在 BIM 技术的应用、参数化设计以及仿真优化三方面[9]。BIM（建筑信息模型）技术的引入，使得钢结构设计从二维图纸向三维模型转变，可实现全专业协同设计，提高设计效率和质量；通过参数化建模，设计师可以根据不同的设计需求快速生成钢结构模型，提高设计的灵活性和效率；通过有限元分析和优化算法，设计师可以对钢结构进行仿真分析，优化设计方案，提高结构的安全性和经济性。

钢结构设计的数字化转型要适配其工业化应用和大跨结构应用[10]，国内许多学者应用数字化技术设计钢结构。陈光在进行城市钢桁架过街天桥设计时使用 MIDAS Civil 建立平面杆件模型，以验算管桁架的动力响应和稳定性[11]，此方法协同了荷载组合和节点设计，但是建模过程繁琐，效率较低；魏勇以南通美术馆为例介绍复杂钢结构的设计过程，采用 ETABS、PKPM、SAP2000 等软件进行整体模型分析和各种补充分析[12]，体现了钢结构性能化设计的理念，但依旧存在重复建模这一机械性工作，导致设计效率低下。

1.2.2 BIM 技术在钢结构设计中的应用

BIM 技术建筑工程中的建筑工程的决策规划、设计、施工、运维过程都有着广泛的应用，但是目前 BIM 在设计阶段中的深度应用较少，大多项目都为施工图翻模，并不能完全发挥出 BIM 的可视性与先进性[13]。

国内外专家应用 BIM 技术进行钢结构设计。陈柳花以宁波市三官堂大桥项目作为案例，介绍了在大跨度钢桁架桥梁设计过程中 TEKLA 软件的详细应用，利用 BIM 技术实现桥梁方案的比选、参数化建模、不同专业之间的模型干涉检查等功能[14]，但并未涉及桁架的内力计算以及节点计算。周勇和屈闫庆以空间坐标为基础数据对钢结构网架进行了参数化建模，并利用空间有限元程序 MIDAS / Gen 进行了钢网架结构的受力计算，其实现了钢结构的受力分析[15]，但并没有将结构设计集成进去，且以节点的坐标为基础数据的建模方式较为复杂，不适用于实际工程。Lai 等人提出了一种集成 Rhino、Grasshopper 和 Abaqus 以建立钢框架的 BIM 模型，并用 SVM 和 NSGA-II 算法优化钢结构的节点[16]，此方法可以有效地辅助设计人员进行钢结构节点设计，但也只是局限于节点设计中，并不能对钢结构的构件进行计算和设计。

虽然国内外专家在钢结构设计阶段中的 BIM 应用开展了大量工作，也取得了重要的进展，但依旧存在难以使用现有的 BIM 软件对钢结构进行正向设计的问题，BIM 与结构设计二者呈分离的状况。

1.3 本文的主要工作

1.3.1 研究内容

本研究针对平面管桁架结构，构建基于 FreeCAD 的参数化 BIM 协同设计系统，依照

“管道-过滤器”架构思维（图 1.1），设计包括参数化建模模块、智能荷载模块、结构设计模块三个程序模块，完成平面管桁架的 BIM 建模、结构分析和结构验算，初步实现对钢结构的正向设计。

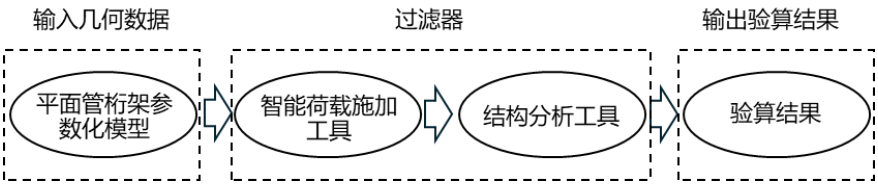


图 1.1 本程序的“管道-过滤器”架构

参数化建模模块是面向桁架拓扑优化的参数驱动引擎，由用户输入设计参数可以即时更新桁架的 BIM 模型；智能荷载模块集成自然语言处理（NLP），支持用户用自然语言输入荷载施加指令，并对荷载进行三维可视化的显示；结构设计模块通过 SAP2000 API 实现内力数据回传，基于《钢结构设计标准》GB50017-2017 对结构构件进行验算。

1.3.2 技术路线

本研究采用“理论分析-工具开发-方案验证”的研究路径（图 1.2）。

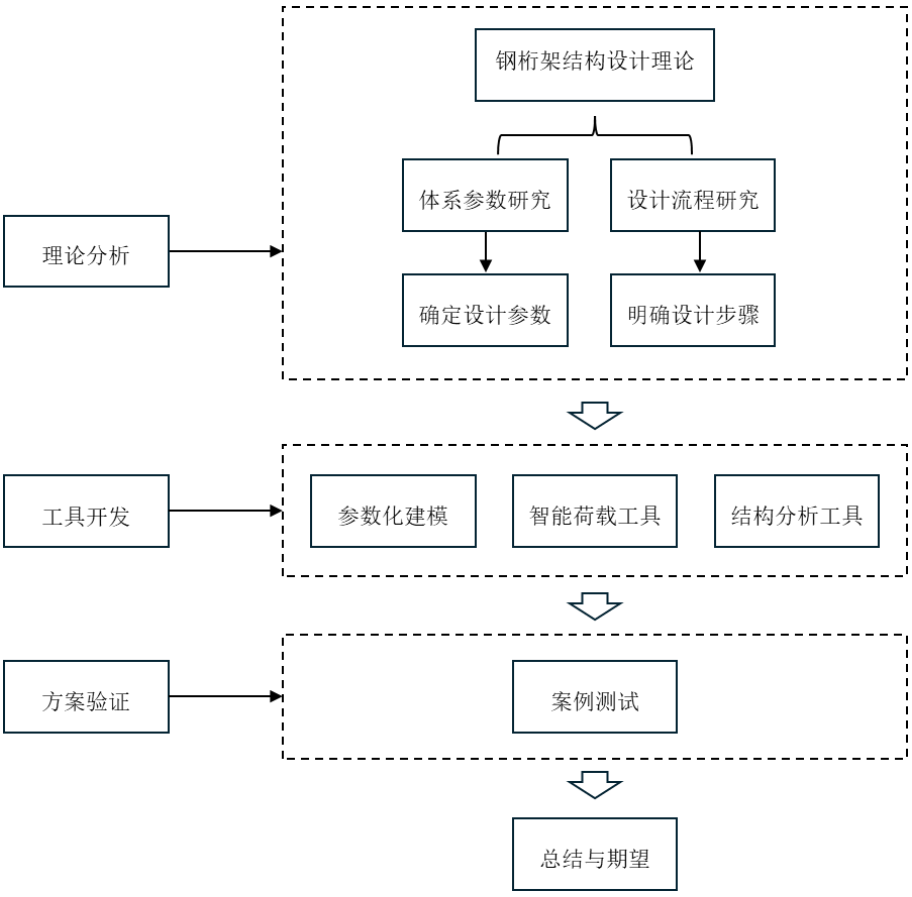


图 1.2 本研究技术路线

2　参数化模型开发

本章通过在 FreeCAD 中定义参数化模型类 ParametricCircularTubeTruss，对平面管桁架的参数化模型进行开发，并实现 FreeCAD 中的 BIM 模型实时更新到 SAP2000 的有限元模型中。几何模型是结构分析和设计的基础，设计者可以借助 BIM 模型更直观的进行结构设计，而结构分析软件（SAP2000）通常需要先建立结构几何才能继续后续分析步骤。参数化模型建模系统提供了这一模型基础。

2.1 系统设计介绍

本研究的参数化建模系统采用分层架构设计，如表 2.1 所，其核心模块包括：用户界面层（User Interface Layer）、业务逻辑层 (Business Logic Layer)、数据层（Data Layer）和外部接口层 (External Interface Layer)。

表 2.1 参数化模型系统的分层架构示意图

用户界面层	FreeCAD GUI 交互
业务逻辑层	参数处理、几何生成算法
数据层	节点、杆件数据结构
外部接口层	SAP2000 API 通信

用户界面层 (User Interface Layer 负责与用户进行交互，直接对接 FreeCAD 的参数化面板，收集设计参数，并采用事件驱动机制实现设计参数的动态响应。该层通过继承 FreeCAD 的 Property 属性体系，构建包含跨度、高度、节间数、杆件截面尺寸等核心参数的交互界面，并借助 onChanged 观察者模式实时捕获参数变更事件。

业务逻辑层 (Business Logic Layer)负责管桁架 BIM 模型的构建，封装了管桁架模型建立的核心逻辑。该层通过 create_nodes 方法实现节点坐标的数学推导，并在 create_members 中依据杆件类型分类实施拓扑连接。

数据层（Data Layer）负责模型信息的存储与管理，通过字典结构精准记录节点的空间坐标，并利用元组键值对建立杆件的连接关系。该层集成 FreeCAD 的文档对象模型（Document Object），通过 obj.addProperty 方法将设计参数持久化存储。

外部接口层 (External Interface Layer)负责 FC 和 SAP2000 之间的协同通信，通过 COM 组件技术实现 FC 与 SAP2000 的无缝对接。该层提供 connect_to_sap2000 连接管理、update_sap2000_model 模型同步等服务，通过 is_sap_connected 状态和异常捕获逻辑（try-except 块）保障跨平台通信的鲁棒性。

2.2 几何模型建立模块

2.2.1 设计参数设定

本模块拟生成参数化平面管桁架的几何模型，设计参数分为几何参数、构造参数和截面参数三类，具体设计参数分类及名称如表 2.2 所示。

表 2.2 平面管桁架模型的设计参数

参数类别	参数名称	数据类型
几何参数	桁架跨度（Span）	浮点数（float）
	桁架高度（Height）	浮点数（float）
	节间数量（Panels）	整数（int）
构造参数	交替斜杆方向（AlternateDiagonals）	布尔（bool）
截面参数	上弦杆直径（TopChordDiameter）	浮点数（float）
	上弦杆壁厚（TopChordThickness）	浮点数（float）
	下弦杆直径（BottomChordDiameter）	浮点数（float）
	下弦杆壁厚（BottomChordThickness）	浮点数（float）
	斜腹杆直径（DiagonalDiameter）	浮点数（float）
	斜腹杆壁厚（DiagonalThickness）	浮点数（float）
	竖腹杆直径（VerticalDiameter）	浮点数（float）
	竖腹杆壁厚（VerticalThickness）	浮点数（float）

通过以上设计参数的设定，可以满足大部分工程的管桁架设计要求。在类 ParametricCircularTubeTruss 中，通过 addProperty 将所有设计参数添加给参数化模型对象（obj），以下代码（图 2.1）为几何参数的添加，构造参数和截面参数的添加与其相类似。

```
    def init_properties(self, obj):
        # 几何属性
        obj.addProperty("App::PropertyLength", "Span", "Truss", "桁架跨度
").Span = 5000.0
        obj.addProperty("App::PropertyLength", "Height", "Truss", "桁架高
度").Height = 1000.0

        obj.addProperty("App::PropertyInteger", "Panels", "Truss", "节间
数量").Panels = 5
```

图 2.1 类 ParametricCircularTubeTruss 中几何参数的添加

2.2.2 节点生成算法

平面管桁架的节点分为上弦节点和下弦节点，节点的生成依赖于几何参数的输入，获取跨度（Span）、高度（Height）和节间数量（Panels）三个几何参数后，通过计算得到节点坐标，并把坐标存储在字典 nodes 中，以便后续生成杆件时的读取。

字典 nodes 的键为节点名称，其命名规则为：从结构的左侧往右计算，上弦节点为 U0、U1 等，下弦节点为 L0、L1 等，名称前面的字母用于区分上下弦节点，数字为递增的编号，如图 2.2 所示。

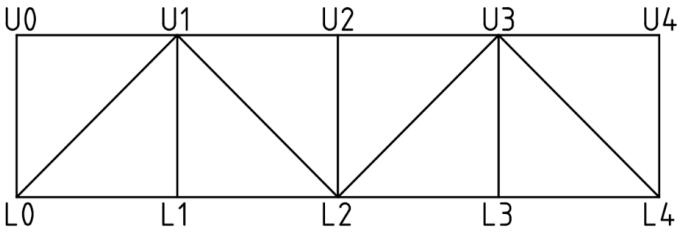


图 2.2 桁架节点命名规则

字典 nodes 的值为节点坐标，数据类型为 Base.Vector，是 FreeCAD 中的三维坐标向量。在计算节点的三维坐标时，结构最左侧的上下弦节点的 x 坐标为 0，向右递增；由于是平面桁架，所有节点的 y 坐标为 0；上弦节点的 z 坐标为桁架高度，下弦节点的 z 坐标为 0。坐标的计算过程如下所示。

上弦节点：$x_{Ui} = i \bullet dx, y_{Ui} = 0, z_{Ui} = Height \quad (i = 0,1,2,...,n)$

下弦节点：$x_{Li} = i \bullet dx, y_{Li} = 0, z_{Li} = 0 \quad (i = 0,1,2,...,n)$

其中，$dx = \dfrac{Span}{Panels}$

2.2.3 杆件拓扑生成

杆件的拓扑生成机制为将相应的节点相连接，杆件生成算法中的核心输入为已得到的节点字典，通过 create_tube 算法将杆件实体对象（Part.Solid）存储到字典 members 中。

字典 members 的键为由构件两端节点名称组成的元组，如上弦构件（U0, U1），斜腹杆（L1, U2），竖腹杆（L3, U3）等，元组第一个字符串是起点，第二个字符串为终点。

字典 members 的值为杆件实体对象（Part.Solid），更具体的说是通过 create_tube 方法生成的空心圆管，此算法需要输入杆件的起点和终点坐标，以及圆管的直径（diameter）和厚度（thickness），通过向量相减得到杆件的方向（direction）和长度（length），再使用 Part 中的 makeCylinder 方法生成直径分别为 diameter 和 diameter – 2* thickness 的两个圆柱体，再执行布尔差集操作，用较小圆柱体切割较大圆柱体，得到所需的空心圆管。以下代码（图 2.3）展示函数 create_tube 的执行过程。

```
1: function CREATE_TUBE(p1, p2, diameter, thickness)
2:   direction ← p2 - p1          ▷ 计算方向向量
3:   length ← direction.magnitude() ▷ 计算两点间距（圆管长度）
4:   direction.normalize()         ▷ 单位化方向向量
5:
6:   outer_radius ← diameter / 2    ▷ 外圆柱半径
7:   outer ← CREATE_CYLINDER(outer_radius, length, p1, direction)
8:
9:   inner_radius ← outer_radius - thickness ▷ 内圆柱半径
10:  inner ← CREATE_CYLINDER(inner_radius, length, p1, direction)
11:
12:  tube ← BOOLEAN_SUBTRACT(outer, inner)    ▷ 外圆柱 - 内圆柱
13:  return tube
14: end function
```

图 2.3 函数 create_tube 执行过程

2.2.4 桁架模型合成

获得杆件字典 members 后，通过定义函数 execute 将所有杆件复合成一个 Part 几何体，将此复合体赋予给参数化模型对象（obj），作为其几何形状（Shape）。

为了方便后续荷载施加等操作，增加了生成可视化节点和清理现有节点两个功能，创建节点对象，方便后续节点识别算法的读取。每次更新参数时调用 execute 函数，除了自动更新模型的几何形状，也会自动更新所有可视化节点。

2.3 SAP2000 实时同步机制

2.3.1 COM 接口通信

COM（Component Object Model） 是一种跨语言、跨进程的二进制接口标准，允许不同软件组件通过标准化协议进行通信。本研究通过 python 的 win32com.client 库调用 SAP2000 API，建立进程间通信，并封装了 connect_to_sap2000，update_sap2000_model 以及 update_sap2000 三个函数分别实现 SAP2000 的连接、有限元模型的创建以及有限元模型的更新。以下代码实现将对象实例化，使用 ProgID：CSI.SAP2000.API.SapObject，通过 Windows 注册表定位 SAP2000 的 COM 服务器，返回的 sap_model 对象提供完整的 API 方法（图 2.4）。

```
self.sap2000 = win32com.client.Dispatch('CSI.SAP2000.API.SapObject')
self.sap_model = self.sap2000.SapModel
```

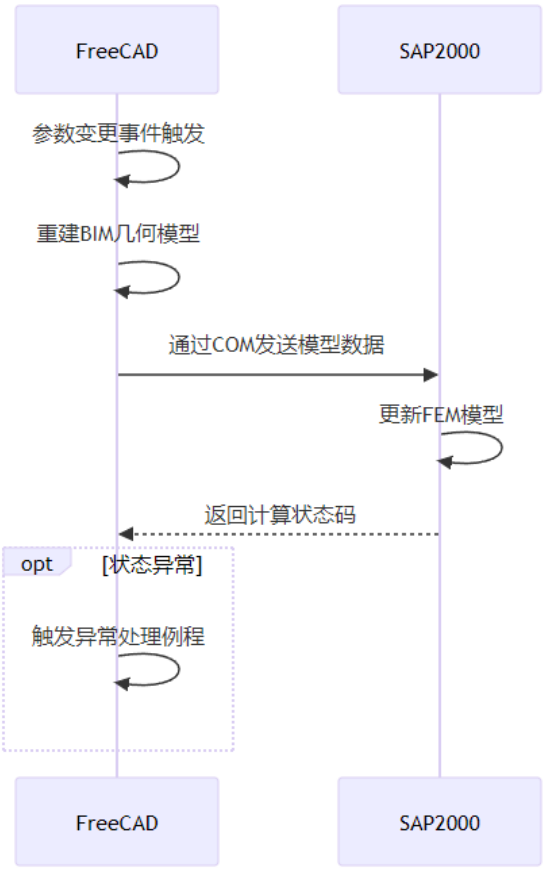


图 2.4 FreeCAD 和 SAP2000 同步机制

2.3.2 属性映射

通过调用 SAP2000 的 API，实现几何、材料、边界条件等多维度数据对齐，保证 BIM 模型与结构分析模型的一致性。

1、 几何属性映射：包括节点坐标和杆件拓扑的映射。节点坐标映射是 FreeCAD 中基于设计

参数生成的节点坐标（如 L0：(0, 0, 0)）通过 PointObj.AddCartesian 写入 SAP2000，此方法所需输入的参数有四个，为 x、y、z 坐标以及节点名称。杆件拓扑映射是利用 FrameObj.AddByPoint 方法将杆件拓扑关系（如上弦杆 U0 - U1）转换为 SAP2000 的框架单元，此方法需输入杆件的起点及终点的名称，从 members 字典中获取杆件的节点信息，确保节点连接顺序一致。

2、 材料属性映射：通过 PropMaterial.SetOSteel 方法，定义钢材的名称、抗拉强度、屈服强度、弹性模量等信息。

3、 截面属性映射：通过 PropFrame.SetPipe 方法，将 FreeCAD 中的设计参数写入 SAP2000，程序依据杆件名称的命名规则，判断杆件类型，以定义上弦杆（TopChord）、下弦杆（BottomChord）、斜腹杆（Diagonal）和竖腹杆（Vertical）的截面几何属性，同时将已定义的材料属性指定到截面属性当中。

4、 边界条件映射：根据桁架结构的计算假定，桁架的所有杆件均为二力杆，即两端铰接的杆件。因此通过 FrameObj.SetReleases 方法，对所有杆件进行端部弯矩的释放。

2.3.3 观察者机制

观察者模式（Observer Pattern）用于实现 FreeCAD 参数修改到 SAP2000 模型更新的事件驱动同步，其核心是通过回调机制监控模型状态变化。

实现流程：首先是事件订阅，在 FreeCAD 中注册参数对象的 onChanged 事件，监听关键属性（如跨度、高度）的修改；若监听到属性的修改，触发 execute 方法重构几何模型，并调用 update_sap2000 推送更新。

3 智能荷载工具

智能荷载工具作为 BIM 模型与结构分析的智能交互层，其功能架构需实现荷载的可视化以及实时同步。本章开发的智能荷载工具可以实现自然语言解析、荷载的三维可视化、荷载数据的映射及实时同步，利用当下流行的 AI 技术，实现荷载加载过程中的自然语言交互，使荷载输入更加简洁便利，大大减小设计意图与工程分析之间的鸿沟。

3.1 系统功能设计

根据智能荷载工具需要实现的功能，将所开发的工具分解为用户交互模块、自然语言处理模块、节点智能识别模块、三维可视化模块以及 SAP2000 同步模块五个核心模块，各个模块的功能以及技术实现如表 3.1 所示。

表 3.1 智能荷载工具的模块划分

模块	功能	技术实现
用户交互模块	提供 GUI 界面（输入框、滑动条、按钮），支持指令输入、参数调节、模型选择	• PySide 控件开发 • 状态反馈机制
自然语言处理模块	解析用户输入的荷载指令，提取荷载类型、大小、方向、作用位置	• DeepSeek API 集成 • 荷载参数映射算法
节点智能识别模块	自动识别上弦节点、支座节点等关键位置	• 坐标阈值算法 • 拓扑关系分析
三维可视化模块	生成动态可调的荷载箭头符号以及支座约束符号	• FreeCAD 建模 API • 动态比例管理器
SAP2000 同步模块	将荷载数据映射至 SAP2000，实现模型荷载的实时更新	• COM 接口通信 • 节点坐标匹配算法

荷载工具系统中的各模块各司其职，共同协作实现基于自然语言的荷载可视化施加。本工具的核心数据流遵循“用户指令-解析-可视化-调整”的闭环逻辑，流程中的每一步由具体模块负责实现，如图 3.1 所示。

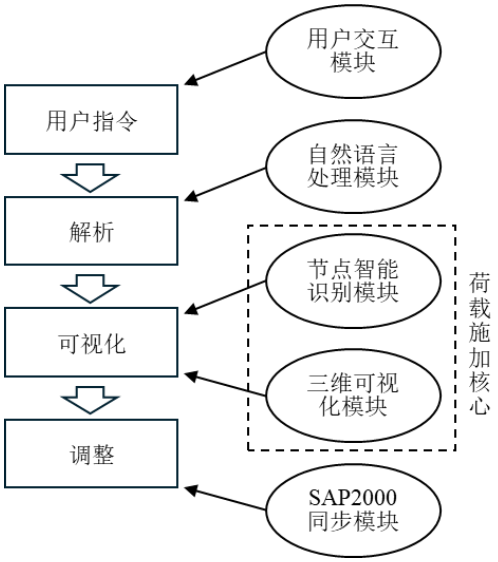


图 3.1 智能荷载工具执行流程

3.2 自然语言处理模块

自然语言处理模块是荷载工具的智能交互核心，实现了从非结构化文本指令到结构化工程参数的自动转换。本模块基于大语言模型（DeepSeek）构建，同时结合专业知识的规则校验数据，定义了类 LoadInstructionParser，开拓更为用户友好的荷载施加方法。

3.2.1 类结构与工作流程

本节提出的自然语言处理模块采用分层封装设计，类命名为 LoadInstructionParser，对 API 通信、语义解析、数据验证等功能进行封装。

1. 属性定义：定义的内容有 API 端点配置、方向映射表和 API 密钥管理。通过常量 API_ENDPOINT 声明了 DeepSeek 的服务地址，建立 DIRECTION_MAP 字典实现方向描述与三维向量的映射关系，在构造函数中接收并存储 API 密钥（api.key）。

2. 方法设计：包括解析主入口（parse 方法）、API 通信层（_call_api 方法）和响应处理层（_format_response 方法）三者。

   ① 解析主入口：带重试机制，采用指数退避策略（最大重试次数=3）应对网络波动，确保程序的鲁棒性。当连续请求失败时，通过 FreeCAD 控制台输出错误日志。

   ② API 通信层：封装 API 请求细节，定义提示词，约束大模型输出为结构化 JSON 格式，同时采用低温系数（temperature = 0.1）降低模型随机性，提升工程指令解析的稳定性。

   ③ 响应处理层：获取 DeepSeek 返回的 JSON 数据，使用 json.loads 进行反序列化，检查必要字段（荷载类型、作用位置、大小、方向）是否缺失，确保数据完整性，最后进行荷载类型、荷载数值、方向描述三项逻辑验证，拦截非法输入。

通过以上的设计，本模块作为自然语言解释器，在接收到用户输入的自然语言文本指令之后，调用 DeepSeek 的 API 向大语言模型发出结构化请求，返回 JSON 格式的荷载指令，

判断 JSON 荷载指令是否涵盖荷载输入所需的必要参数，若荷载指令满足要求，则向存储并输出此标准化指令，再通过三维可视化模块和 SAP2000 同步模块执行后续的荷载可视化和同步操作。其完整的工作流程如图 3.2 所示。

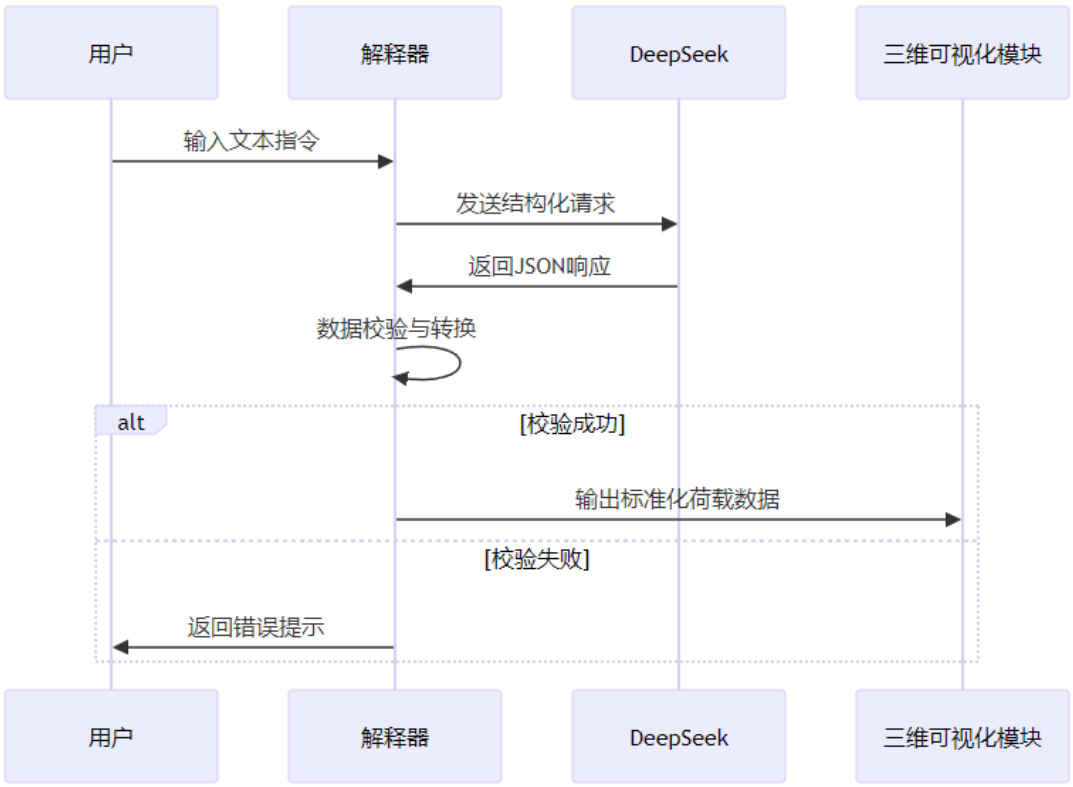


图 3.2 自然语言处理解释器工作机制

3.2.2 大语言模型集成

大语言模型（Large Language Models, LLMs）是基于海量文本数据预训练的超大规模语言模型，其主要能力是通过深度学习技术理解并生成类人文本，具有深度语义理解、知识密集型推理等特征[17]。DeepSeek 是一家位于中国杭州的创新型科技公司，推出的 AI 模型 DeepSeek R1 引发全球关注，其核心突破在于以极低成本开发出与顶级大语言模型（如 ChatGPT-o1）性能相当的模型，同时采用开源策略，显著降低了开发者的使用门槛。[18]

在本模块中，选择接入 DeepSeek（DS），主要原因有：

1. 优秀的推理能力：DS 性能强大，能够在短时间内完成复杂的推理任务，完全可以胜任本模块中荷载指令的解析任务。
2. 中文指令优势：DS 的训练数据包含大量中文语料，对中文文本指令的歧义消解能力更强。
3. 使用成本较低：相对其它大语言模型，DS 的使用成本较低，使用 deepseek-chat 模型的百万 tokens 输出价格仅 8 元人民币，不到 ChatGPT 使用成本的一半（2.7 美元，约 18 人民币）。

通过系统提示词引导大语言模型输出结构化 JSON 数据，在设计提示词时，包括荷载的

四要素（大小、方向、作用位置、荷载类型）。大小是以千牛（kN）为单位的数值；方向是大语言模型可以识别的方向描述中文字符串；荷载类型初步设为“活”和“恒”两种类型。

荷载作用位置的解析较为复杂，桁架结构的荷载是假定作用在节点之上的，只需判断荷载指令企图对哪些节点施加，即提取出节点的序号，以数字的形式存储于 value 中；但考虑到用户在时对节点位置的描述具有多样性，还需判别用户对节点的描述方法，如“全部节点”、“第 2、5 个节点”、“前 5 个节点”、“x 坐标大于 5m 的节点”，设定了三种描述类型（type），分别为“all”、“indexes”和“condition”，再针对各种描述类型定义节点提取方法。通过以上的解析，将作用位置指令转化成结构为（type, value）的元组。

另外，通过“作为结构工程专家”为大语言模型指派专家身份，确保更好完成荷载参数提取的任务，具体的提示词代码如图 3.3 所示。

```
system_prompt = """作为结构工程专家，请从以下指令提取荷载参数，按JSON格式返回：
{
    "load_type": "恒/活",
    "position": {"type": "all/indexes/range/condition", "value": [...]},
    "magnitude": 数值（kN）,
    "direction": "方向描述"
}"""
```

图 3.3 提示词工程

在参数配置方面，选用“deepseek-chat”模型，强制输出 JSON 格式，同时设定温度参数（temperature）为 0.1，抑制输出的随机性，增加超时控制（timeout）为 100 秒，如图 3.4。

温度参数（temperature）是控制大语言模型生成文本随机性的核心超参数，在本模块中选用温度参数 0.1，可以降低输出的随机性，抑制模型对数值的模糊化表达，同时保障结构化的输出。例如，确保荷载类型、方向描述等匹配预设术语，避免歧义，防止荷载值出现如“约 20kN”的模糊数值，减少 JSON 格式错误，提高解析成功率。

超时控制（timeout）是网络请求的最长等待时间，timeout＝100 意味着若服务器在 100 秒内未返回响应，则自动终止请求。参数 timeout 过长可能导致程序界面卡顿，影响用户使用体验；过短则可能导致解析请求被过早终止，无法获得解析结果。

经过参数调优，（temperature＝0.1，timeout＝100）可以取得在响应速度和可靠性之间的平衡，使自然语言处理模块在效率、准确性和稳定性上满足使用要求。

```
return requests.post(
    self.API_ENDPOINT,
    headers=headers,
    json={
        "model": "deepseek-chat",
        "messages": [
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": text}
        ],
        "response_format": {"type": "json_object"},
        "temperature": 0.1
    },
    timeout=100  # 增加超时控制
)
```

图 3.4 调用 DeepSeek 的参数配置

3.2.3 数据验证机制

获得 DeepSeek 返回的 JSON 格式数据后，通过混合验证机制对数据进行校验，同时对方向数据进行转化。

1. 数据完整性校验：检查返回的字段是否完整包含荷载的四要素（'load_type', 'position', 'magnitude', 'direction'）。
2. 荷载类型检查：检查荷载类型是否为“恒”或“活，保证荷载类型有效。
3. 荷载数值检查：确保荷载数值大于 0，转化为以千牛为单位的数值大小。
4. 方向向量转化：检查返回的方向数据是否有效，并实现方向数据向三维向量的映射。通过字典 DIRECTION_MAP 存储标准方向向量（竖直向下、竖直向上、水平向左、水平向右等），检查获取的方向描述是否存在于字典 DIRECTION_MAP 中，将方向描述转化为三维向量。

本模块得到的 JSON 格式数据，可以精确反映用户输入的荷载施加自然语言，实现了从非结构化文本到可执行荷载参数的精确转换。

3.3 节点识别与三维可视化

节点智能识别模块和三维可视化模块是荷载工具的执行核心，两个模块协作完成荷载的可视化符号创建以及施加定位（图 3.5），本节将详细解释这两个模块的技术实现算法。

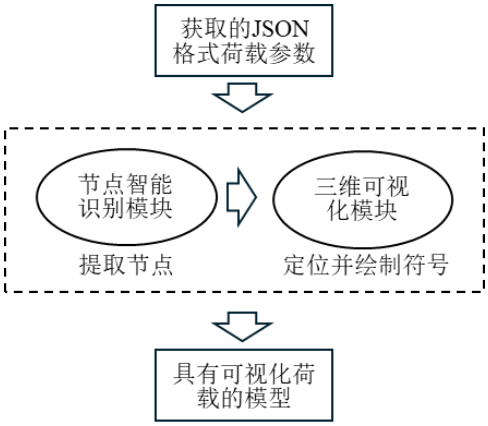


图 3.5 节点识别和可视化模块的功能

3.3.1 节点智能识别算法

为实现荷载的精准施加，需从模型中提取关键节点，本模块创建了两个节点识别器，分别实现上弦节点和支座节点的识别提取，上弦节点的提取保证了荷载施加的加载基础，而支座节点的提取则为程序提供了设置支座的位置。

上弦节点识别器的工作原理：遍历 FreeCAD 文档中所有对象，筛选出顶点对象（Vertexes），存储于列表 nodes 中。获得列表 nodes 后，遍历 nodes 中所有顶点，提取顶点对象的 z 坐标最大值（z_max）和最小值（z_min），计算出高度阈值（threshold），高度阈值的计算公式为：

$$threshold = z_{\min} + (z_{\max} - z_{\min}) \times ratio$$

其中 ratio 为阈值比例参数，在上弦节点识别时设定为 0.7。高度阈值的物理意义是获取 z 坐标高于高度阈值的节点，即认为高于阈值的节点为上弦节点。高度阈值的设定可以使上弦节点的识别更具有普适性，适用于各种形式的桁架模型，例如上弦杆具有坡度的，若坡度较大时，则应适当调低比例参数（ratio）。

通过高度阈值筛选的节点为候选节点，存储于列表 candidates 中，以节点对象的 x 坐标为排序条件，获得排序过后上弦节点，并存储在键为节点名称，值为节点对象的字典中。

支座节点识别器的工作原理类似于上弦节点识别器，同样是通过高度阈值区分上下弦节点，设定比例参数 ratio 为 0.3，得到下弦节点，再排序节点，最后选取首末两个节点对象为支座节点。

3.3.2 荷载与支座的定位

荷载定位指的是根据 JSON 格式的荷载施加指令找到荷载所在的节点，在自然语言处理模块当中，用户的自然语言指令已经被转化为 JSON 格式的荷载指令，荷载作用位置已被解析成结构为（type, value）的元组。使用节点识别算法读入上弦节点后，通过_select_nodes 方法来选择荷载指令选定的节点，此方法中根据荷载描述方法（type）分类，定义了索引选择器（_select_by_index）和条件选择器（_filter_by_condition）两个节点选择器以处理不同描述

方法的荷载指令。

若荷载描述方法为“all”，则选择所有上弦节点；若为“indexes”或“condition”，则调用自定义的节点选择器（图 3.6）。

索引选择器：将荷载指令中的节点索引从 1-based 索引（如 1 表示第一个节点）转换为 Python 的 0-based 索引，同对索引进行过滤，滤除非整数、数值小于 0 以及数值超出节点个数的索引。

条件选择器：使用了 python 内置的 eval 函数进行节点筛选，eval 函数的作用为解析字符串，其完整语法为 eval(source, globals=None, locals=None)，其中 source 是需解析的字符串，globals 和 locals 分别为全局变量空间和局部变量空间，均为字典形式。在条件选择器中，source 为变量 condition，即经过自然语言处理模块解析后的荷载施加位置描述，如“X>10m”，表示在 x 坐标大于 10m 的节点施加荷载；全局变量空间和局部变量空间限制了 eval 函数的执行环境，限制其可访问的变量为节点的 x、y、z 坐标，防止出现安全问题。

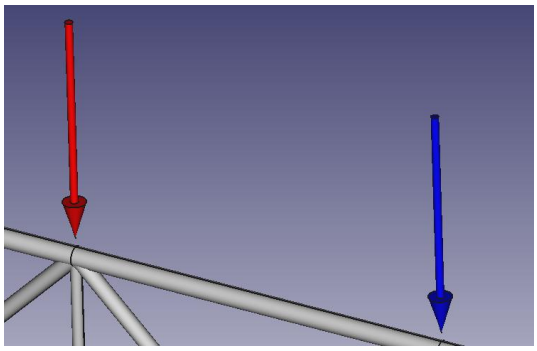
```
def _select_by_index(self, nodes, indexes):
    """索引选择器（1-based转0-based）"""
    valid_indexes = [i-1 for i in indexes if isinstance(i, int) and i > 0]
    return [nodes[i] for i in valid_indexes if i < len(nodes)]

def _filter_by_condition(self, nodes, condition):
    """安全条件筛选器"""
    filtered = []
    for node in nodes:
        point = node.Shape.Vertexes[0].Point
        try:
            # 使用限制环境执行eval
            if eval(condition, {'X': point.x, 'Y': point.y, 'Z': point.z}, {}):
                filtered.append(node)
        except:
            continue
    return filtered
```

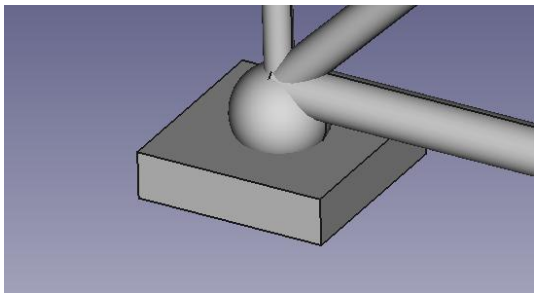
图 3.6 节点选择器的代码

3.3.3 三维符号设计与绘制

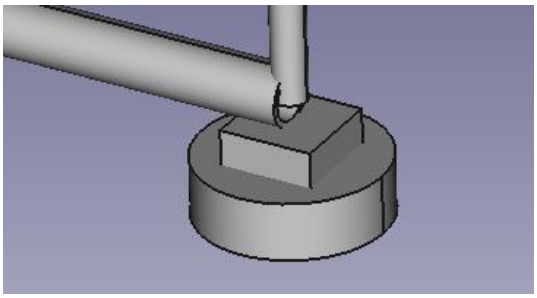
本节将对荷载符号和支座符号进行设计和绘制。荷载符号设计为三维的箭头，箭头指向代表力的方向，该三维符号由圆锥和圆柱组合而成，根据荷载类型分为两种颜色，红色表示恒载，蓝色表示活载，如图 3.7-a 所示；支座符号分为铰支座和固定支座，铰支座设计为球体和立方体的叠加（图 3.7-b），用球体表示支座处的弯曲和扭转不受约束，固定支座设计为圆柱体和立方体的叠加（图 3.7-c）。



(a) 荷载符号



(b) 铰支座



(c) 固定支座

图 3.7 三维可视化符号

1. 荷载符号的绘制

在设计的工具中，荷载被表示为三维箭头，本节定义了两个函数（_create_shaft 和 _create_head）来分别绘制三维箭头的杆部和头部。通过使用 FreeCAD 文档的 addObject 添加箭头杆部和头部的实例对象，分别为圆柱（Part::Cylinder）和圆锥（Part::Cone），在两个函数中分别指定圆柱、半径的半径和高度，完成荷载符号部件的创建。

荷载符号的部件创建完成后，使用函数（_create_arrow）对部件进行组合，并对符号参数完成参数化。首先是位置的读取，将自然语言处理模块中解析出的加载节点导入；其次是方向的读取，确保荷载符号指向指定的方向；最后是荷载符号长度的设置，通过比例常数（scale_factor）设定符号与实际荷载值的对应关系，如当 scale_factor = 50，则表示 1kN 的荷载在 FreeCAD 中表示为 50mm 长的三维箭头。

$$
\begin{aligned}
&l = m \times 50 \\
&R_{shaft} = 5, H_{shaft} = l \\
&R_{head} = 3 \times 5 \times l, H_{head} = 0.2 \times l
\end{aligned}
$$

创建复合对象（Part::Compound），使用 Links 命令对荷载符号部件进行组合。为区分恒载和活载两种荷载类型，对组合后的荷载符号进行命名和颜色设置。荷载符号对象的名称为“荷载类型+荷载值”，如“恒 10kN”；为使荷载类型的区分更加直观，使用 ViewObject.ShapeColor 命令，将恒载符号设置为红色，活载符号设置为蓝色。

2. 箭头比例管理器

为让所创建的荷载符号在不同尺度的模型中保持可视性，新定义了荷载符号比例管理器

（ArrowScaleManager 类），采用单例模式（Singleton），其功能为统一管理所有箭头的比例，并且实时同步更新。

单例模式是指在比例管理器中只创建一个实例，其主要目的为保持全局状态的一致性，确保全局只有一个 ArrowScaleManager 实例，统一管理所有箭头的缩放比例。管理器实例包含全局比例因子（scale_factor）和箭头信息字典（arrow_objects）两个重要参数，全局比例因子控制箭头的缩放，箭头信息字典存储所有已注册箭头的信息。

将新创建的箭头注册到管理器中，即将箭头所代表荷载的部分荷载信息（荷载值、荷载方向、作用位置）存储到箭头信息中（字典 arrow_objects）。

在更新荷载符号大小时时，所有箭头同步更新。函数 update_all 的更新逻辑是修改比例因子，检查并清除箭头信息字典存在的已被删除的箭头对象，再调用函数_update_single，对确认存在并注册的箭头进行参数更新。

函数_update_single 的更新算法并不复杂，即获取 FreeCAD 文档中存在的箭头对象，在箭头杆部和头部的计算参数中，乘入一个比例因子，完成箭头大小的调节更新。如下面公式所示，m 为箭头所代表的荷载值，scale 为比例因子。

$$
\begin{aligned}
&l_{new} = m \times 50 \times scale \\
&R_{shaft} = 5 \times scale, H_{shaft} = l_{new} \\
&R_{head} = 3 \times 5 \times l_{new}, H_{head} = 0.2 \times l_{new}
\end{aligned}
$$

3. 支座符号的绘制

铰支座由基础和旋转球体构成。基础为方块对象（Part::Box），传入长、宽、高以及位置四个参数；旋转球体为球体对象（Part::Sphere），传入半径和位置两个参数。最后通过复合对象（Part::Compound）的 Links 方法将二者组合。

固定支座由基础和固定块构成，基础是圆柱体对象（Part::Cylinder），固定块为方块对象（Part::Box），与铰支座类似，创建完成基础和固定块后，将二者组合为复合对象（Part::Compound）。

3.4 用户交互与 SAP2000 同步

3.4.1 用户交互界面设计

本工具通过调用 PySide 构建了三级控制界面以实现用户交互，包括工具栏、对话框和浮动面板三类组件，如表 3.2 所示。

表 3.2 用户交互组件分类

组件类型	主要功能	技术实现
工具栏	功能入口集群	FreeCADGui.Workbench 扩展机制

对话框	荷载指令输入、支座选择	QDialog 子类化设计
浮动面板	实时比例调节	QSlider 动态控制

工具栏的创建是通过继承 FreeCADGui.Command 类和 FreeCADGui.Workbench 类实现的。首先定义命令（Command），设计 Activated()（点击时的操作）和 GetResources()（定义图标、菜单文本），接入工具其它模块的功能，之后注册命令；在工作台集成时，继承 FreeCADGui.WorkbenchInitialize()，在 Initialize()方法中添加命令、菜单和工具栏。

本工具定义并注册了 SAP 文件选取、添加支座、施加荷载和实时比例四个命令，设计了四个按钮，集成到工具栏中，如图 3.8 所示。

图 3.8 荷载工具的工具栏

对话框的创建是通过继承 QtWidgets.QDialog 类实现的。在_init_ui()方法中组织对话框的控件，_on_apply()方法处理用户的输入并完成后续的命令。以荷载指令输入对话框为例，在界面布局设计（_init_ui）时采用垂直布局（QVBoxLayout）组织控件，创建输入框、按钮和状态标签（图 3.9）；在业务逻辑处理（_on_apply）方法中，调用自然语言处理模块对输入进行解析，之后调用荷载施加可视化模块，根据荷载指令进行荷载施加，并返回和显示荷载施加状态。

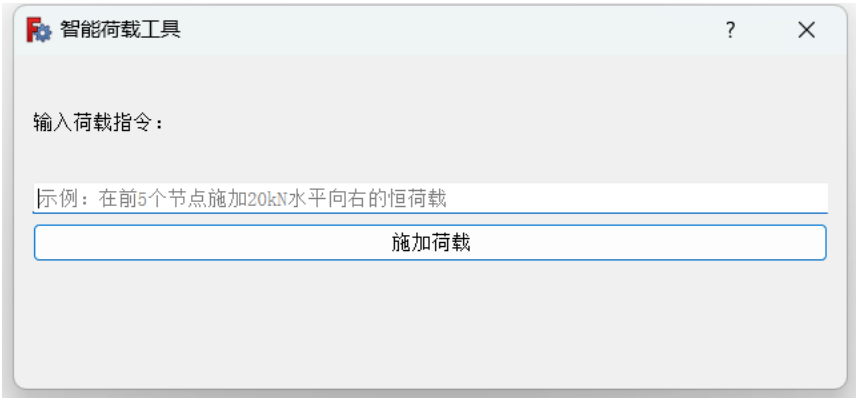


图 3.9 荷载指令输入对话框

本工具中浮动面板（图 3.10）的功能是对箭头比例进行实时调节，利用 QSlider 实现，通过 QtWidgets.QVBoxLayout()设置水平滑动条，并将比例传递到箭头比例管理器中。

图 3.10 比例调节浮动面板

3.4.2 SAP2000 同步机制

采用 COM 接口实现跨平台数据通信，为实现荷载的同步，设计了节点匹配算法（find_matching_node）和 SAP2000 连接器（SAP2000Connector）。在同步荷载时，先通过 SAP2000 连接器与 sap 文件实现连接，读取节点坐标，使用坐标匹配算法找到施加荷载的节点位置，若节点匹配成功，则调用 SAP200 API 进行荷载定义、施加，最后自动保存 sap 文件，其同步流程如图 3.11 所示。

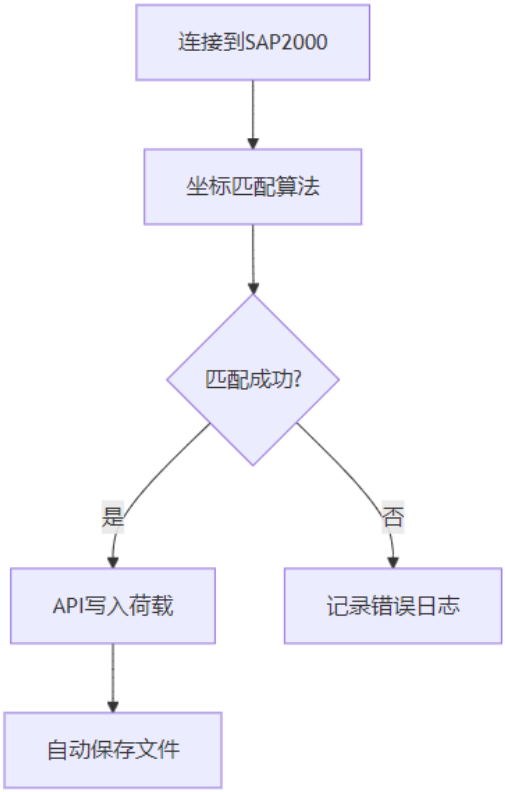


图 3.11 荷载的 SAP2000 同步流程

1. 节点匹配算法

节点匹配算法的实现是对比节点坐标差，若差值小于容许值则匹配成功，返回该节点名称。具体来说是先创建一个列表，存储 SAP 文档中的所有节点名称和坐标，遍历存储 SAP

文档节点的列表，通过节点名称找到与需要匹配的 FC 文档节点，进行坐标对比，验证坐标是否匹配。

2. SAP2000 连接器

为实现 FreeCAD 与 SAP2000 的安全连接，设计了 SAP2000 连接器（SAP2000Connector），该连接器集成了从模型加载、交互操作到异常处理的全流程支持，在本工具开发中作为一个相对独立的模块。

交互式的文件选择，集成 QFileDialog 提供文件选择界面，获取 SAP 文件的路径；SAP2000 模型连接管理，加载模型文件，检查文件是否存在和格式是否达标，通过 COM 接口启动 SAP2000 进程并获取模型对象。

调用 SAP2000 API 写入荷载，使用 PointObj.SetLoadForce 方法，并自动保存模型。

4　结构分析工具

FreeCAD 仅为 BIM 建模软件，并不能对结构进行有限元分析，选用的结构分析软件 SAP2000 可以实现对结构的分析，但无法直接输出结构验算结果。本工具的可以实现在 FreeCAD 界面中读取结构模型，并完成桁架结构的分析验算，以表格的形式展示验算结果。

4.1 系统架构设计

结构分析工具采用模块化架构，将程序分为用户交互模块、数据加载模块、构件分析模块和 SAP2000 连接模块四个模块，各个模块的功能和技术实现如表 4.1 所示。

表 4.1 结构分析工具各模块及其功能

模块	功能	技术实现
用户交互模块	提供 GUI 界面，支持结构数据、验算结果以表格形式展示	• PySide 控件开发 • 状态反馈机制
数据加载模块	获取节点、杆件的几何、构造和内力等信息，并标准化存储至 FC 文档	• 字典树结构存储 • Json 序列化
构件分析模块	根据《钢结构设计标准》GB50017-2017 实现强度、刚度、稳定性验算	• 数据建模@dataclass • 规范公式（math）
SAP2000 连接模块	用户交互式加载 SAP2000 模型文件	• COM 接口通信

分析工具中的各模块各司其职，模块间的交互关系如图 4.1 所示。

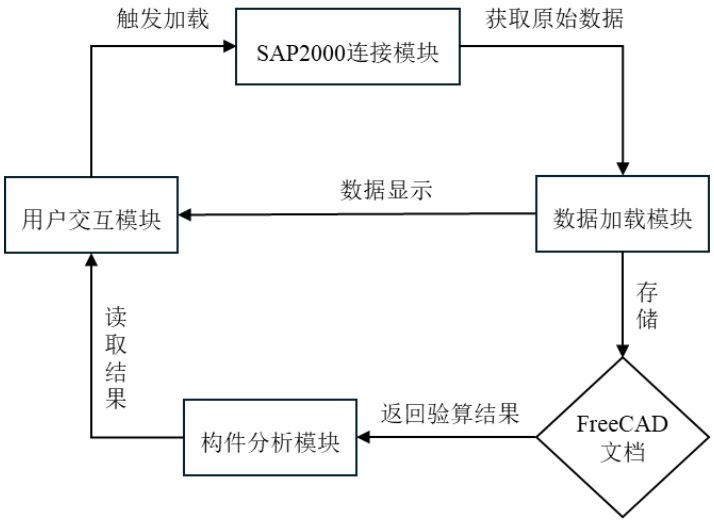


图 4.1 结构分析工具中各模块交互关系

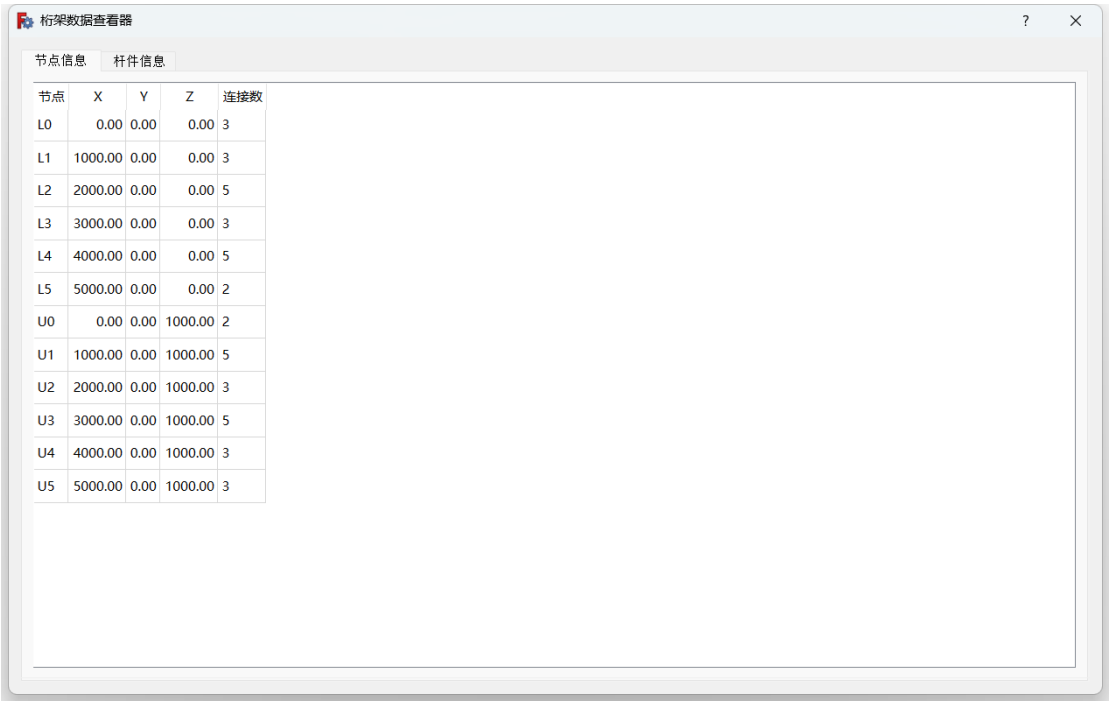
4.2 用户交互与 SAP2000 连接

4.2.1 用户交互模块

用户交互模块的主要功能是触发程序的运行，并且查看得到的结果。用户交互模块的设计原理与荷载工具的用户交互模块相同，通过 FreeCADGui.Workbench 创建工具栏，注册“选择 SAP 模型”、“加载桁架数据”和“验算构件”三个功能（图 4.2）；调用 PySide 库创建两个分别包含结构数据和验算数据的可视化表格。

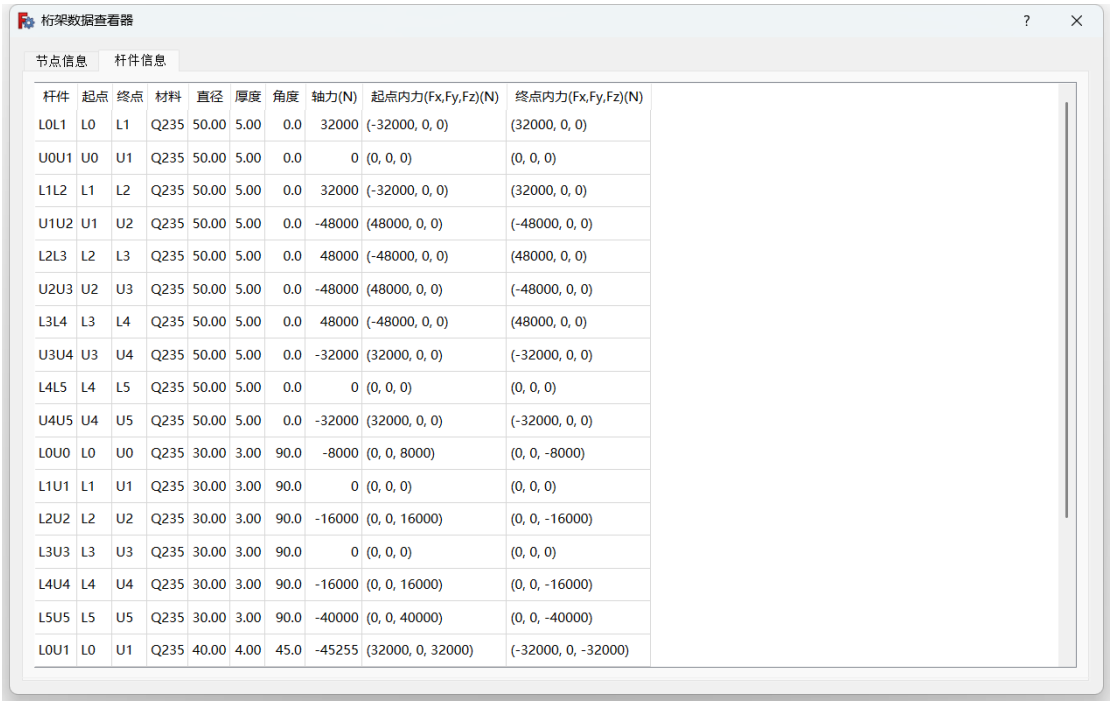
图 4.2 结构分析工具的工具栏

可视化表格是继承 QDialog，创建了对话框，并且使用 QTable 作为基础组件来创建表格。结构数据的表格是一个双标签页的表格，第一个标签页展现的是节点的数据（图 4.3），第二个标签页展现构件的数据（图 4.4）；验算数据表格显示的是每根构件的强度、刚度、稳定性验算的实际值和限值（图 4.5）。



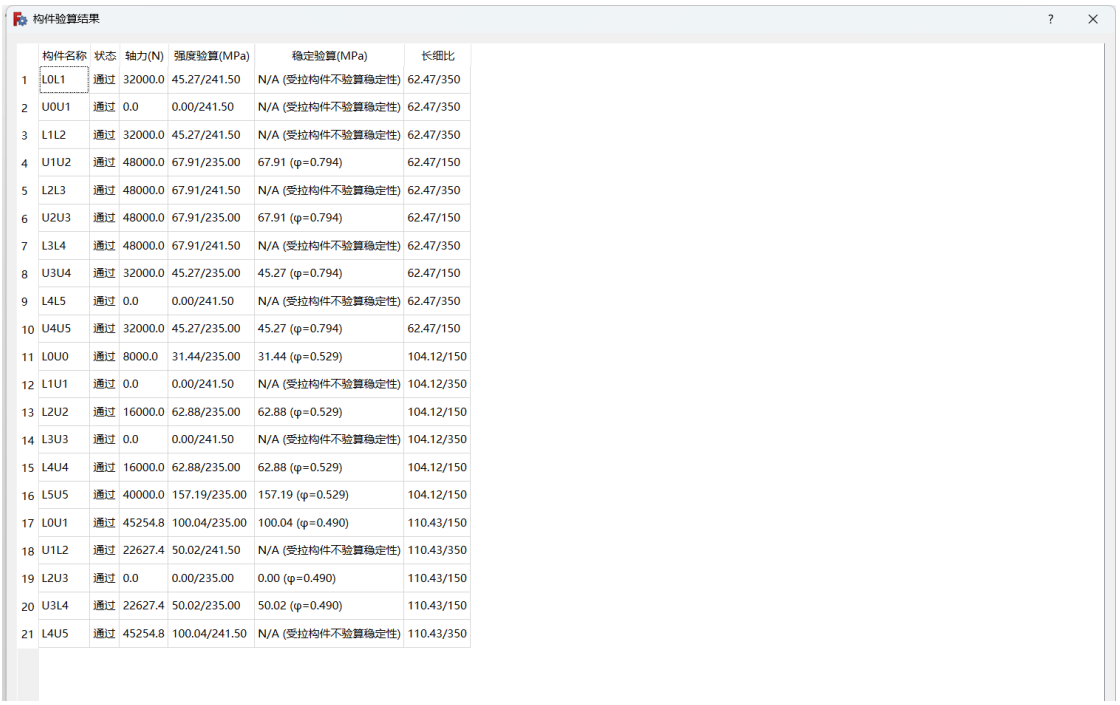
节点	X	Y	Z	连接数
L0	0.00	0.00	0.00	3
L1	1000.00	0.00	0.00	3
L2	2000.00	0.00	0.00	5
L3	3000.00	0.00	0.00	3
L4	4000.00	0.00	0.00	5
L5	5000.00	0.00	0.00	2
U0	0.00	0.00	1000.00	2
U1	1000.00	0.00	1000.00	5
U2	2000.00	0.00	1000.00	3
U3	3000.00	0.00	1000.00	5
U4	4000.00	0.00	1000.00	3
U5	5000.00	0.00	1000.00	3

图 4.3 节点数据表格



杆件	起点	终点	材料	直径	厚度	角度	轴力(N)	起点内力(Fx,Fy,Fz)(N)	终点内力(Fx,Fy,Fz)(N)
L0L1	L0	L1	Q235	50.00	5.00	0.0	32000	(-32000, 0, 0)	(32000, 0, 0)
U0U1	U0	U1	Q235	50.00	5.00	0.0	0	(0, 0, 0)	(0, 0, 0)
L1L2	L1	L2	Q235	50.00	5.00	0.0	32000	(-32000, 0, 0)	(32000, 0, 0)
U1U2	U1	U2	Q235	50.00	5.00	0.0	-48000	(48000, 0, 0)	(-48000, 0, 0)
L2L3	L2	L3	Q235	50.00	5.00	0.0	48000	(-48000, 0, 0)	(48000, 0, 0)
U2U3	U2	U3	Q235	50.00	5.00	0.0	-48000	(48000, 0, 0)	(-48000, 0, 0)
L3L4	L3	L4	Q235	50.00	5.00	0.0	48000	(-48000, 0, 0)	(48000, 0, 0)
U3U4	U3	U4	Q235	50.00	5.00	0.0	-32000	(32000, 0, 0)	(-32000, 0, 0)
L4L5	L4	L5	Q235	50.00	5.00	0.0	0	(0, 0, 0)	(0, 0, 0)
U4U5	U4	U5	Q235	50.00	5.00	0.0	-32000	(32000, 0, 0)	(-32000, 0, 0)
L0U0	L0	U0	Q235	30.00	3.00	90.0	-8000	(0, 0, 8000)	(0, 0, -8000)
L1U1	L1	U1	Q235	30.00	3.00	90.0	0	(0, 0, 0)	(0, 0, 0)
L2U2	L2	U2	Q235	30.00	3.00	90.0	-16000	(0, 0, 16000)	(0, 0, -16000)
L3U3	L3	U3	Q235	30.00	3.00	90.0	0	(0, 0, 0)	(0, 0, 0)
L4U4	L4	U4	Q235	30.00	3.00	90.0	-16000	(0, 0, 16000)	(0, 0, -16000)
L5U5	L5	U5	Q235	30.00	3.00	90.0	-40000	(0, 0, 40000)	(0, 0, -40000)
L0U1	L0	U1	Q235	40.00	4.00	45.0	-45255	(32000, 0, 32000)	(-32000, 0, -32000)

图 4.4 构件数据表格



	构件名称	状态	轴力(N)	强度验算(MPa)	稳定验算(MPa)	长细比
1	L0L1	通过	32000.0	45.27/241.50	N/A (受拉构件不验算稳定性)	62.47/350
2	U0U1	通过	0.0	0.00/241.50	N/A (受拉构件不验算稳定性)	62.47/350
3	L1L2	通过	32000.0	45.27/241.50	N/A (受拉构件不验算稳定性)	62.47/350
4	U1U2	通过	48000.0	67.91/235.00	67.91 (φ=0.794)	62.47/150
5	L2L3	通过	48000.0	67.91/241.50	N/A (受拉构件不验算稳定性)	62.47/350
6	U2U3	通过	48000.0	67.91/235.00	67.91 (φ=0.794)	62.47/150
7	L3L4	通过	48000.0	67.91/241.50	N/A (受拉构件不验算稳定性)	62.47/350
8	U3U4	通过	32000.0	45.27/235.00	45.27 (φ=0.794)	62.47/150
9	L4L5	通过	0.0	0.00/241.50	N/A (受拉构件不验算稳定性)	62.47/350
10	U4U5	通过	32000.0	45.27/235.00	45.27 (φ=0.794)	62.47/150
11	L0U0	通过	8000.0	31.44/235.00	31.44 (φ=0.529)	104.12/150
12	L1U1	通过	0.0	0.00/241.50	N/A (受拉构件不验算稳定性)	104.12/350
13	L2U2	通过	16000.0	62.88/235.00	62.88 (φ=0.529)	104.12/150
14	L3U3	通过	0.0	0.00/241.50	N/A (受拉构件不验算稳定性)	104.12/350
15	L4U4	通过	16000.0	62.88/235.00	62.88 (φ=0.529)	104.12/150
16	L5U5	通过	40000.0	157.19/235.00	157.19 (φ=0.529)	104.12/150
17	L0U1	通过	45254.8	100.04/235.00	100.04 (φ=0.490)	110.43/150
18	U1L2	通过	22627.4	50.02/241.50	N/A (受拉构件不验算稳定性)	110.43/350
19	L2U3	通过	0.0	0.00/235.00	0.00 (φ=0.490)	110.43/150
20	U3L4	通过	22627.4	50.02/235.00	50.02 (φ=0.490)	110.43/150
21	L4U5	通过	45254.8	100.04/241.50	N/A (受拉构件不验算稳定性)	110.43/350

图 4.5 验算结果表格

4.2.2 SAP2000 连接器

FreeCAD 文档需要先和 SAP2000 结构模型连接，才能获取结构的数据。本工具沿用了在荷载工具中设计的 SAP2000 连接器，集成 QFileDialog，提供用户交互式选择 SAP2000 结构模型的服务（图 4.6），并通过 COM 接口启动 SAP2000 进程并获取该被选定的结构模型，完成与 SAP2000 的连接。

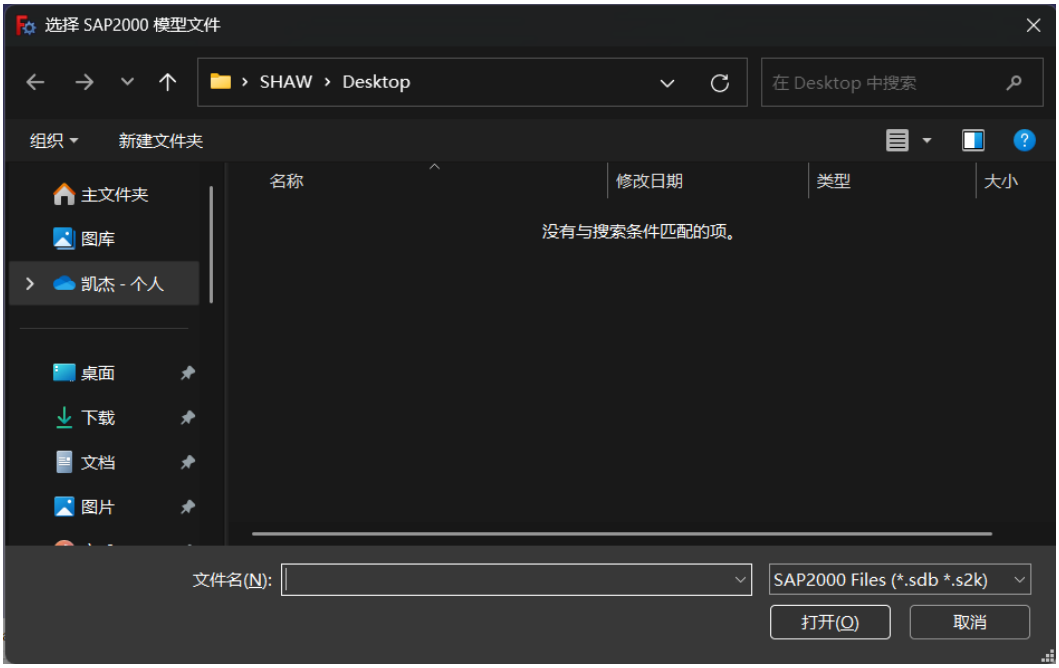


图 4.6 交互式选择 SAP 模型文件

4.3 结构数据加载模块

结构数据加载模块的功能是获取桁架结构模型中节点、杆件的信息。具体而言，通过调用 SAP2000 API，提取了桁架结构模型节点的名称、三维坐标和相连构件数量，模型构件的名称、长度、材料、截面尺寸以及轴力和起点、终点的节点力。同时将以上数据存储到 FreeCAD 文档中，以便后续对结构进行验算时读取。

4.3.1 数据获取

调用 SAP2000 API 获得桁架结构的各项数据。本模块定义了“_load_nodes”和“_load_frames”两个函数，分别实现节点数据和构件数据的获取，又定义了“_get_section”和“_get_force”以获取构件数据中的截面尺寸和轴力、节点力数据。

在节点加载函数（_load_nodes）中，调用 SAP2000 的 PointObj.GetNameList 命令，得到结构模型中所有节点的名称并存储于列表中，再遍历节点名称列表，调用 PointObj.GetCoordCartesian 命令获取所有节点的坐标。为得到节点所连接的构件，通过 FrameObj.GetNameList 获得所有构件名称并存于列表，遍历构件名称列表，调用 FrameObj.GetPoints 读取每根构件的起点、终点，分别以起点、终点为键，构件为值，放入字典中，最终可以得到形如{ node1: [frame1, frame2] }的字典，表示节点 node1 所连接的构件是 fram1 和 frame2，通过 len() 函数得到 node1 所连接构件数量为 2 个。

构件加载函数（_load_frames）的加载过程与加载节点的过程类似。通过 FrameObj.GetNameList 得到构件名称列表，得到构件起点和终点后，根据坐标使用距离公式计算出构件的长度；自定义的函数“_get_section”可以得到构件的材料和截面尺寸，具体是调用了 PropFrame.GetPipe 命令，获得构件截面的信息列表，通过列表索引提出所需的数据；构件的轴力和节点力的获取分别是通过调用 Results.FrameForce 和 Results.FrameJointForce 命令实现的，在获取构件的力数据前需要先对 SAP2000 模型进行分析参数设置，即单位制

和分析工况的设置，将这一完整的力数据获取流程打包成函数“_get_force”。

4.3.2 数据存储

在 4.3.1 中阐述了本模块从 SAP2000 获取的过程，但没有详细说明这些的数据是如何存储的。本模块采用树形字典存储数据，即母字典下方嵌套若干子字典，其数据结构如图 4.7 所示，具体代码如下。

```
data = {
    'nodes': nodes[name] = { 'x': x, 'y': y,'z': z, 'frames': [ ] },
    'frames': frames[name] = { 'nodes': [node1, node2], 'section': section, 'length': length, 'angle': angle },
    'forces': force_data = { 'frame_forces': {}, 'joint_forces': {} }
    }
```

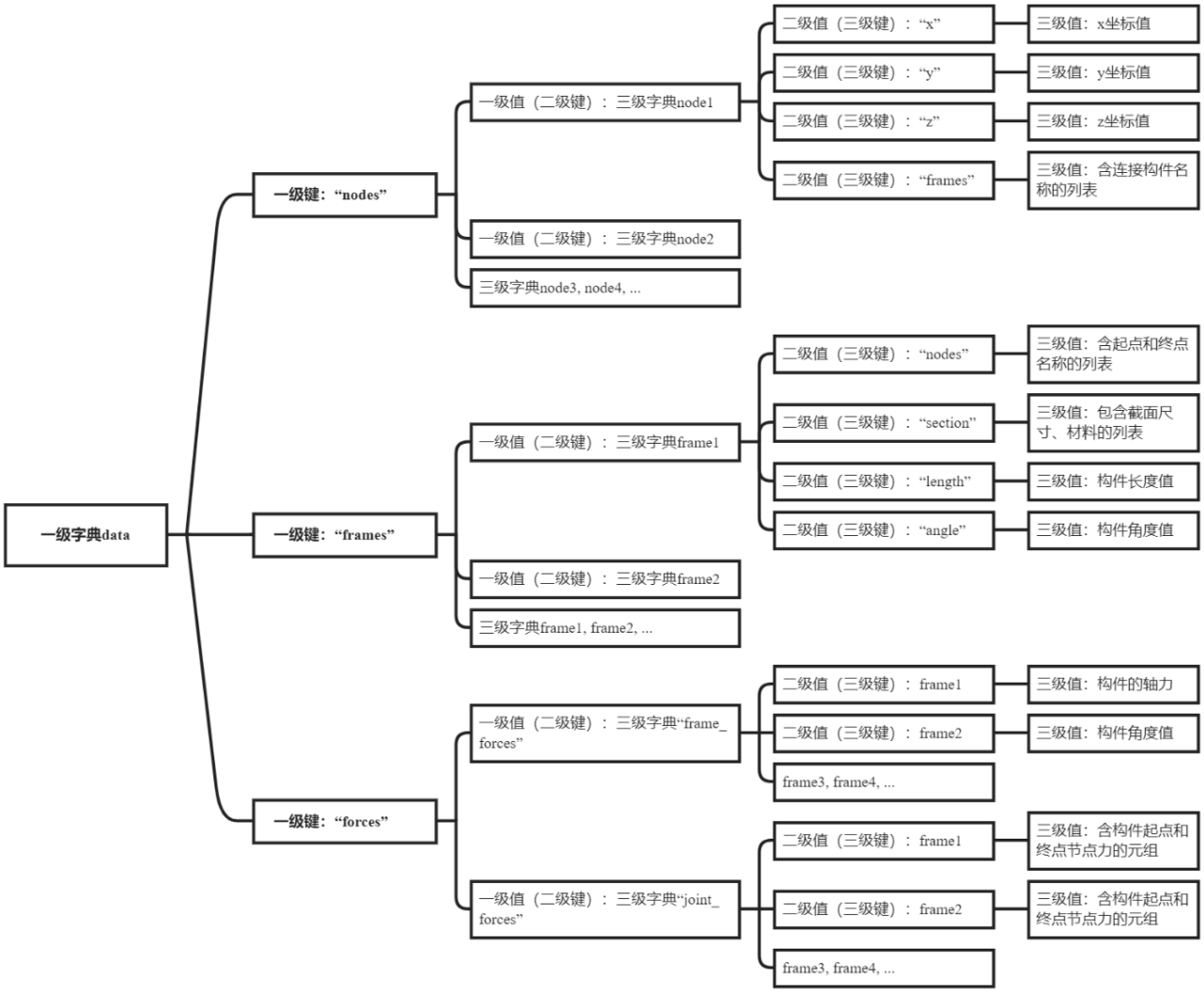


图 4.7 树形字典的结构示意图

本模块的树形节点总共嵌套了三级字典，一级字典 data 是存储最外层的母字典，含有三个键“nodes”、“frames”和“forces”，它们均为二级字典，分别存储着节点、构件和力的数据；二级字典“nodes”的键是若干个三级字典 node，三级字典 node 的键为“x”、“y”、“z”和“frames”，分别存储着节点的 x、y、z 坐标和所连接构件的名称；二级字典“frames”的键是若干个三级字典 frame，三级字典 frame 的键为“nodes”、“section”、“length”和“angle”，它们分别存储着所连接节点名称、截面尺寸和材料、构件长度以及构件放置角度；二级字典“forces”的键为“frame_forces”和“joint_forces”，这二者为三级字典，它们的键为模型中的构件名称，值分别为轴力和节点力。

4.4 构件分析验算模块

构件分析验算模块针对钢桁架结构的构件进行验算，通过数据建模将离散的设计参数进行系统化管理，再读取这些设计参数，对轴心受力构件进行强度、刚度和稳定性三级验算，将验算结果进行存储和管理。

4.4.1 数据建模体系

本模块使用@dataclass 装饰器构建数据模型，定义了四个嵌套类（_HollowSection、_SteelMaterial、_MemberForce 和_MemberGeometry），实现对构件截面参数、钢材材料参数、构件内力和构件几何参数的系统化管理。

数据建模方式相对传统的字典存储更为简洁，并且通过__post_init__函数可以实现参数自校验，如对于圆管截面参数，其壁厚不应大于圆管半径，且所有尺寸参数为整数，在数据建模类中通过以下代码实现参数自校验（图 4.8）。

```python
@dataclass
class _HollowSection:
    """空心圆管截面参数"""
    D: float  # 外径(mm)
    t: float  # 壁厚(mm)

    def __post_init__(self):
        """数据有效性验证"""
        if self.t >= self.D / 2:
            raise ValueError("壁厚不能超过半径")
        if self.D <= 0 or self.t <= 0:
            raise ValueError("尺寸必须为正数")
```

图 4.8 计算参数自校验

4.4.2 三级验算体系

集成《钢结构设计标准》GB50017-2017 对构件进行强度、刚度、稳定性验算。

在验算前，通过解析几何参数计算关键截面属性，根据外径 D 和壁厚 t，调用 math 库编写内径 d、截面面积 A、惯性矩 I 和回转半径 i 三个关键属性。计算公式如下所示：

$$d = D - 2t$$
$$A = \pi \bullet (D^2 - d^2)/4$$
$$I = \pi \bullet (D^4 - d^4)/64$$
$$i = \sqrt{I/A}$$

轴心受力构件的强度应满足：

$$\sigma = \frac{N}{A_n} \le f \text{ 。}$$

其中：当构件受压时，f 取钢材屈服强度 f_y；当构件受拉时，f 取 $0.7f_u$

刚度校验即对构件的长细比进行验算，根据 GB50017-2017，对于受压构件，$\lambda \le 150$；对于受拉构件，$\lambda \le 350$。

钢圆管截面为 b 类截面，进行稳定性验算时，稳定系数 φ 计算公式如下：

$$\varphi = \begin{cases} 1 - \alpha_1 \lambda_n^2, \lambda_n \le 0.215 \\ \frac{1}{2\lambda_n^2}[(\alpha_2 + \alpha_3 \lambda_n + \lambda_n^2) - \sqrt{(\alpha_2 + \alpha_3 \lambda_n + \lambda_n^2) - 4\lambda_n^2}], \lambda_n > 0.215 \end{cases}$$

将以上计算过程分别包装成函数“_verify_strength”、“_verify_slenderness”和“_verify_stability”，完成对强度、刚度、稳定性的验算。

4.4.3 验算结果管理

用字典 results 存储验算结果，以便再用户交互模块中以表格形式展示结果，字典结构如下所示。

```
result = { 'status': "通过",   # 整体状态
    'checks': [ { 'name': '强度验算',
                  'value': 215.3,   # 计算值
                  'limit': 235,     # 允许值
                  'passed': True,   # 是否通过
                  'φ': 0.923        # 稳定系数(可选)
                }, …  ],
  'N_Ed': 120450   # 轴力设计值(N)
```

5　工程案例应用

选取某钢屋架，使用本研究所开发的工具对其进行建模和分析验算。本章将展示本研究所开发的工具的使用过程，并对其得到的结果与手算结果进行对比。

5.1 设计信息

某钢屋架采取平行弦桁架形式，其简支在钢筋混凝土柱顶上，柱顶标高为 9.000 m，柱截面为 400 mm × 400 mm，用混凝土强度等级为 C30。屋架无天窗，外形尺寸如图 5.1 所示。

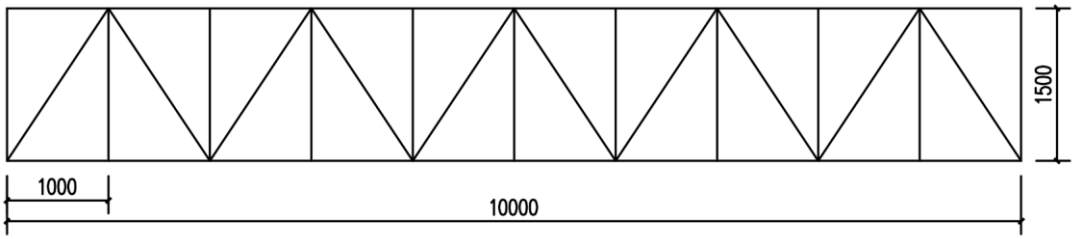


图 5.1 某钢屋架外形尺寸图

屋架的构件选用钢圆管截面，上弦杆截面选用 $\phi 100\times 6$，下弦杆截面选用 $\phi 90\times 4$，竖腹杆和斜腹杆截面均选用 $\phi 70\times 3$。经过荷载组合，在最不利荷载作用下，屋架中间节点所受的节点力为 20 kN，端部节点为 10 kN。

5.2 模型生成

根据屋架的设计信息，修改参数化模型的跨度、高度、节间数以及截面信息（图 5.2），生成 FreeCAD 中的 BIM 模型和 SAP2000 中的有限元模型（图 5.3、图 5.4）。

Property	Value
Label	Truss
Bottom Chord	
Bottom Ch...	90.00 mm
Bottom Ch...	4.00 mm
Diagonals	
Diagonal D...	70.00 mm
Diagonal T...	3.00 mm
Top Chord	
Top Chord...	100.00 mm
Top Chord...	6.00 mm
Truss	
Alternate ...	true
Height	1500.00 mm
Nodes	[TrussNode022 (L011), TrussNode023 (L0...
Panels	10
Span	10.00 m
Verticals	
Vertical Di...	70.00 mm
Vertical Thi...	3.00 mm

图 5.2 修改参数化模型的设计参数

根据修改过后的设计参数，很快便生成了对应的 BIM 模型和有限元模型，此时还未给桁架模型添加支座。

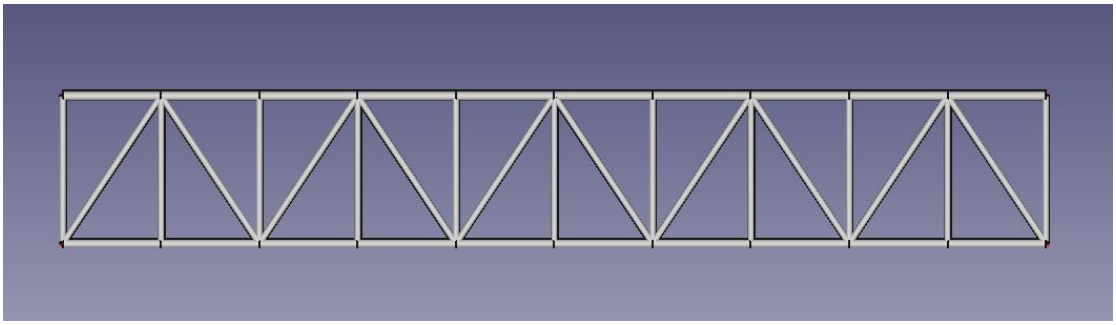


图 5.3 在 FreeCAD 中生成的 BIM 模型

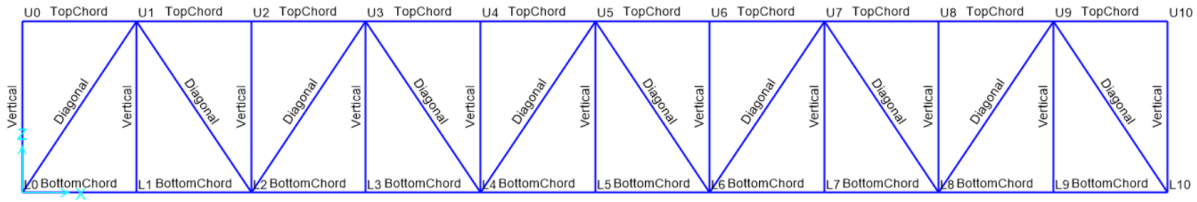


图 5.4 在 SAP 中同步生成的有限元模型

5.3 荷载施加

生成 BIM 模型和有限元模型后，进行支座的布置和荷载的施加。打开“智能荷载根据”，点击“添加支座”，为桁架两端布置铰支座（图 5.5，图 5.6）。

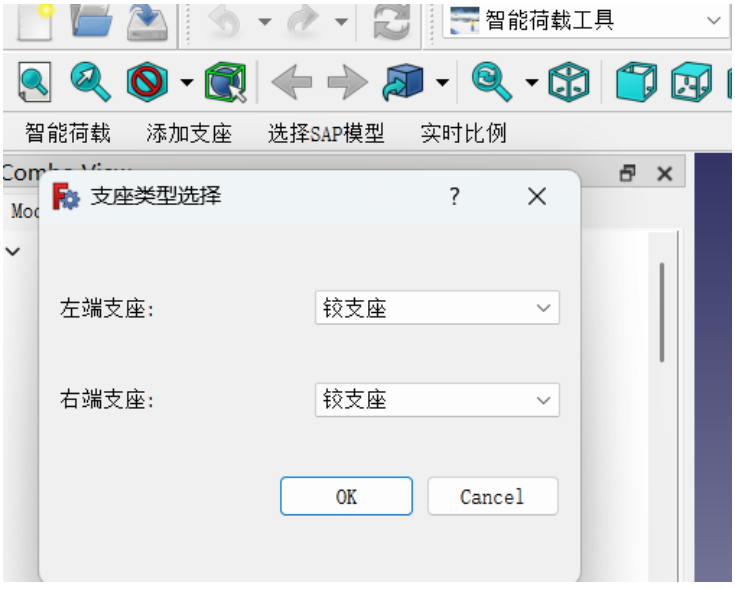


图 5.5 支座布置界面

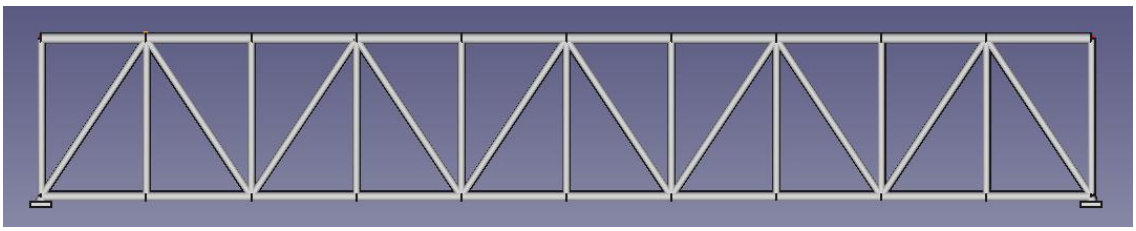


图 5.6 具有支座的 BIM 模型

支座布置完成后，点击“选择 SAP 模型”，选取在 5.2 生成的 SAP 有限元模型，选取完成后点击“智能荷载”，按照设计信息进行荷载施加，由于本荷载工具暂时不提供荷载组合的功能，在此按恒荷载施加设计信息中的节点力，中间节点 20 kN，端部节点 10 kN。

在自然语言对话框中依次输入“在第 2、3、4、5、6、7、8、9、10 个节点施加 20kN 竖直向下的恒荷载”和“在第 1 和第 11 节点施加 10kN 竖直向下的恒荷载”，如图 5.7、图 5.8 所示。等待程序响应，完成 BIM 模型的荷载施加和可视化（图 5.9），同步完成 SAP 有限元模型中的荷载布置（图 5.10）。

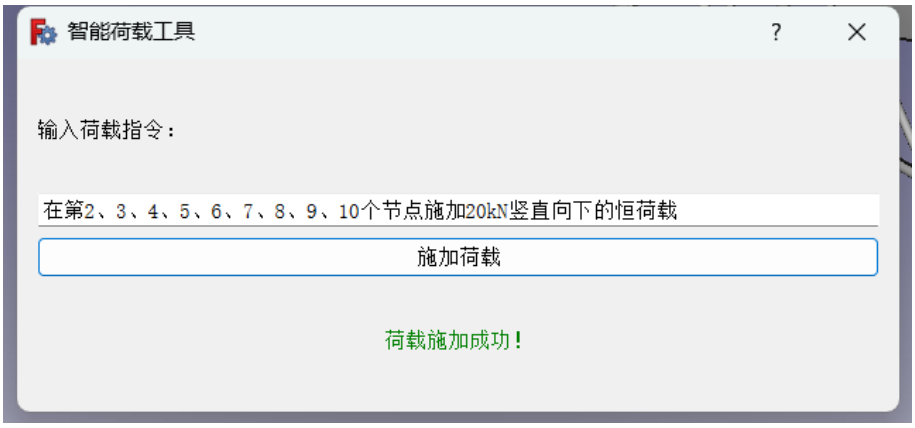


图 5.7 施加中间节点的节点力

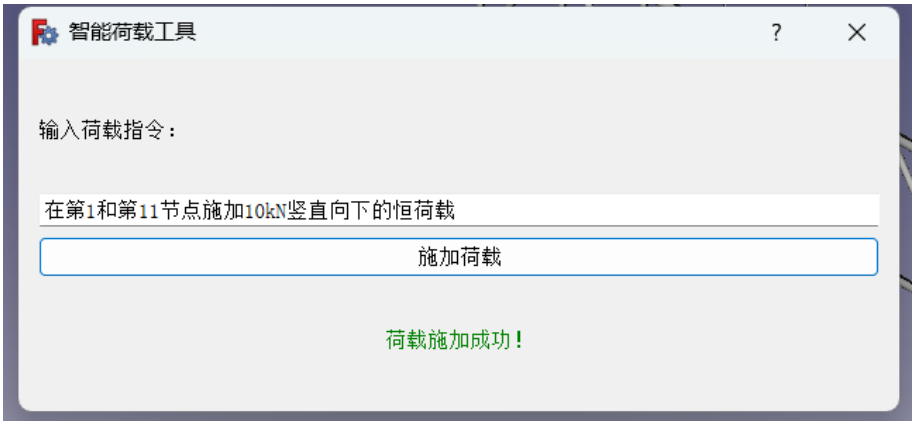


图 5.8 施加端部节点的节点力

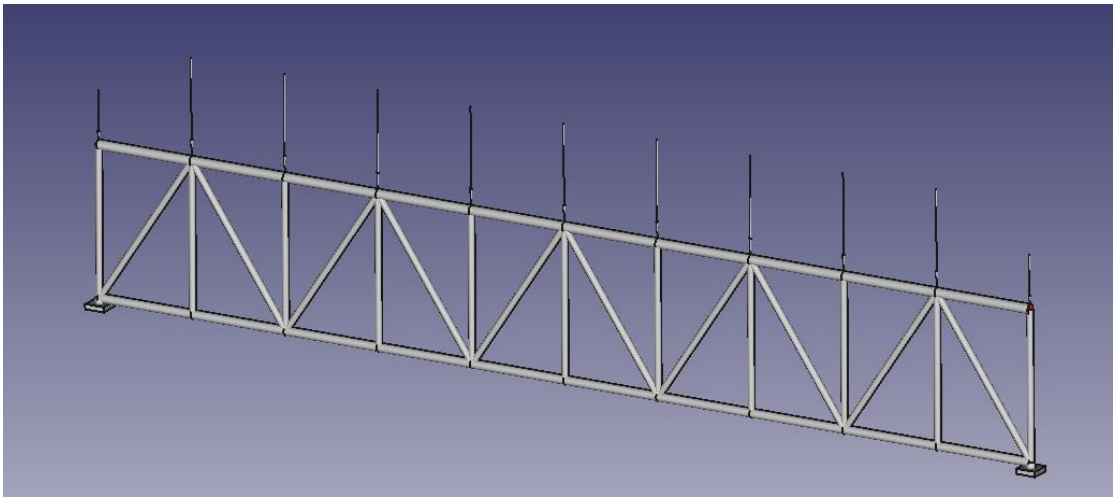


图 5.9 完成可视化荷载的施加

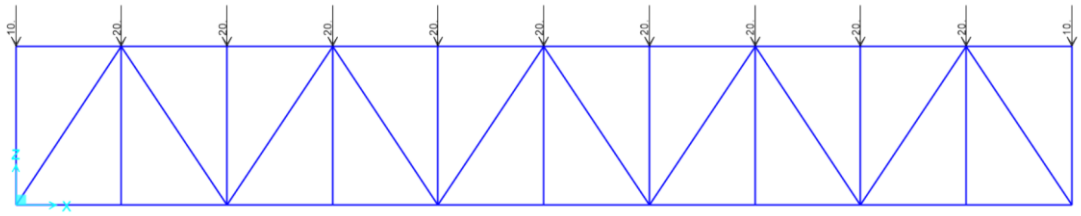


图 5.10 荷载同步施加到 SAP 有限元模型

在 BIM 模型中生成的荷载箭头过小，显示不够清晰，使用“实时比例”调节箭头的大小（图 5.11），使荷载显示更加直观。

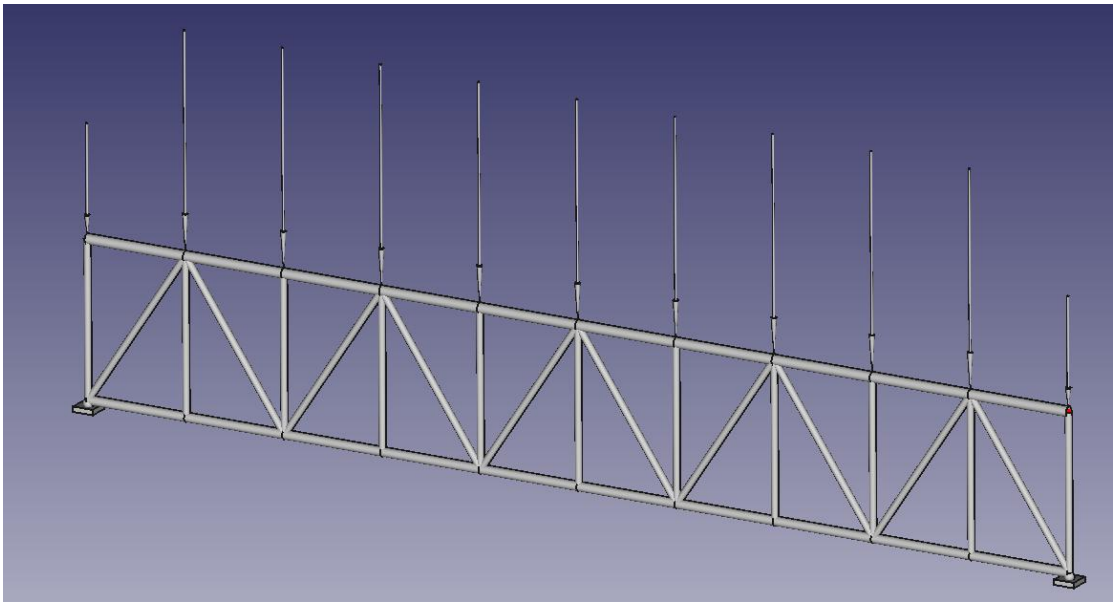


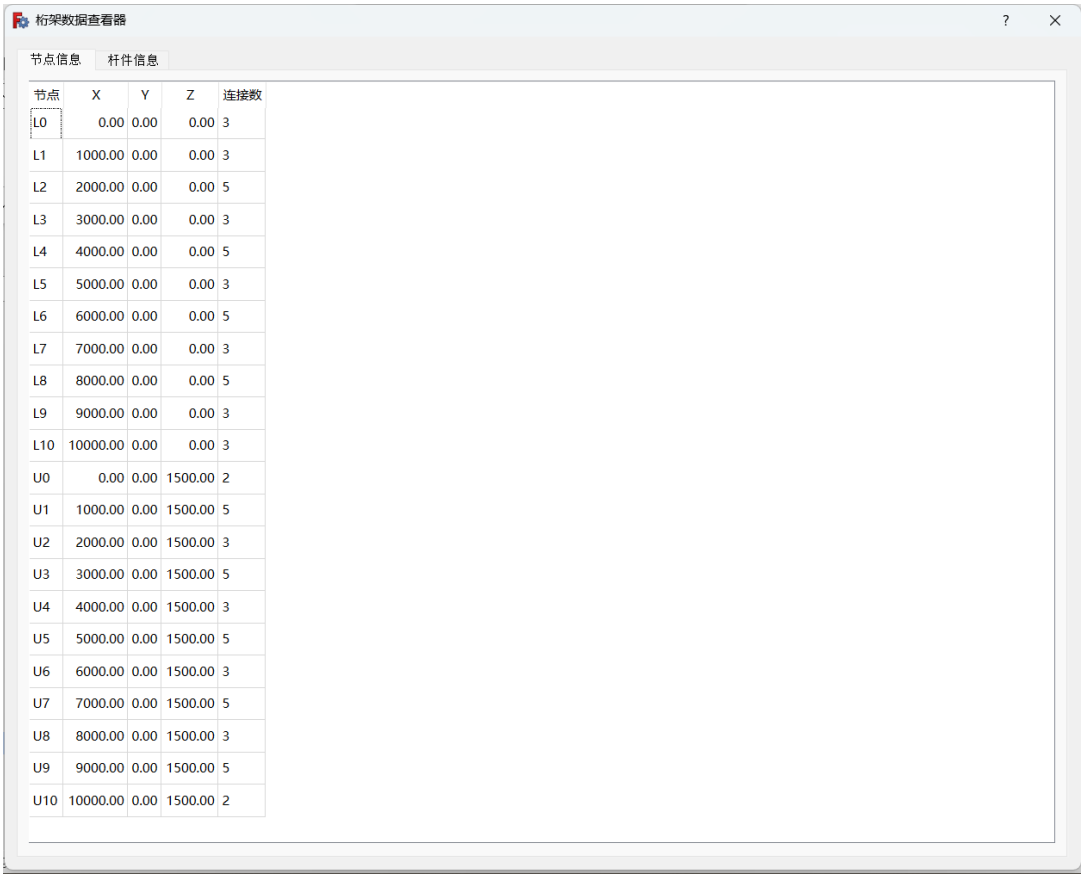
图 5.11 荷载箭头的调节

通过以上步骤，完成了荷载的在 BIM 模型中的可视化以及有限元模型中的施加。

5.4 构件验算

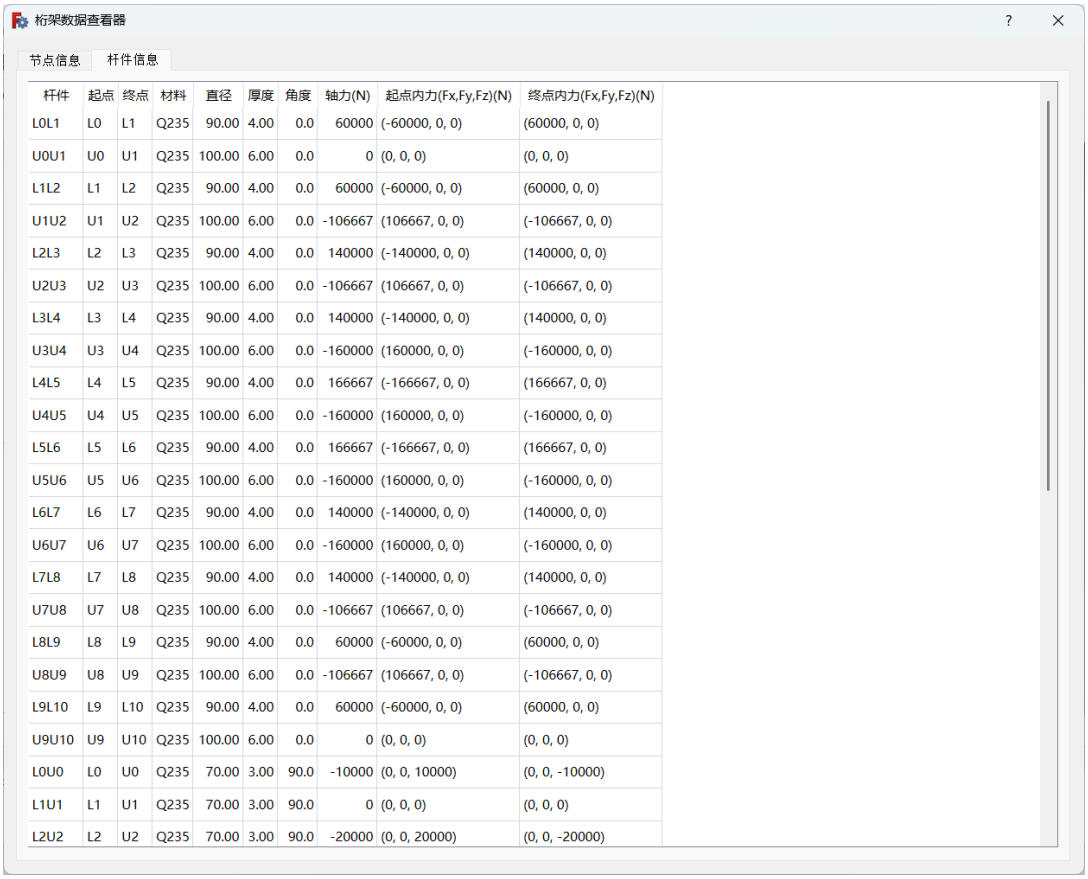
完成荷载布置后，对桁架模型进行力学分析，并验算其所有构件。打开“结构分析工具”，先点击“选择 SAP 模型”加载有限元模型，点击 “加载桁架数据”，完成桁架结构模型的信息获取（图 5.12，图 5.13）。

关闭结构信息的表格，点击“验算构件”，程序调用 SAP2000 对桁架结构进行力学分析，并且通过分析工具对所有构件依次进行验算，得到所有构件的验算结果（图 5.14）。



节点	X	Y	Z	连接数
L0	0.00	0.00	0.00	3
L1	1000.00	0.00	0.00	3
L2	2000.00	0.00	0.00	5
L3	3000.00	0.00	0.00	3
L4	4000.00	0.00	0.00	5
L5	5000.00	0.00	0.00	3
L6	6000.00	0.00	0.00	5
L7	7000.00	0.00	0.00	3
L8	8000.00	0.00	0.00	5
L9	9000.00	0.00	0.00	3
L10	10000.00	0.00	0.00	3
U0	0.00	0.00	1500.00	2
U1	1000.00	0.00	1500.00	5
U2	2000.00	0.00	1500.00	3
U3	3000.00	0.00	1500.00	5
U4	4000.00	0.00	1500.00	3
U5	5000.00	0.00	1500.00	5
U6	6000.00	0.00	1500.00	3
U7	7000.00	0.00	1500.00	5
U8	8000.00	0.00	1500.00	3
U9	9000.00	0.00	1500.00	5
U10	10000.00	0.00	1500.00	2

图 5.12 桁架模型的节点信息



杆件	起点	终点	材料	直径	厚度	角度	轴力(N)	起点内力(Fx,Fy,Fz)(N)	终点内力(Fx,Fy,Fz)(N)
L0L1	L0	L1	Q235	90.00	4.00	0.0	60000	(-60000, 0, 0)	(60000, 0, 0)
U0U1	U0	U1	Q235	100.00	6.00	0.0	0	(0, 0, 0)	(0, 0, 0)
L1L2	L1	L2	Q235	90.00	4.00	0.0	60000	(-60000, 0, 0)	(60000, 0, 0)
U1U2	U1	U2	Q235	100.00	6.00	0.0	-106667	(106667, 0, 0)	(-106667, 0, 0)
L2L3	L2	L3	Q235	90.00	4.00	0.0	140000	(-140000, 0, 0)	(140000, 0, 0)
U2U3	U2	U3	Q235	100.00	6.00	0.0	-106667	(106667, 0, 0)	(-106667, 0, 0)
L3L4	L3	L4	Q235	90.00	4.00	0.0	140000	(-140000, 0, 0)	(140000, 0, 0)
U3U4	U3	U4	Q235	100.00	6.00	0.0	-160000	(160000, 0, 0)	(-160000, 0, 0)
L4L5	L4	L5	Q235	90.00	4.00	0.0	166667	(-166667, 0, 0)	(166667, 0, 0)
U4U5	U4	U5	Q235	100.00	6.00	0.0	-160000	(160000, 0, 0)	(-160000, 0, 0)
L5L6	L5	L6	Q235	90.00	4.00	0.0	166667	(-166667, 0, 0)	(166667, 0, 0)
U5U6	U5	U6	Q235	100.00	6.00	0.0	-160000	(160000, 0, 0)	(-160000, 0, 0)
L6L7	L6	L7	Q235	90.00	4.00	0.0	140000	(-140000, 0, 0)	(140000, 0, 0)
U6U7	U6	U7	Q235	100.00	6.00	0.0	-160000	(160000, 0, 0)	(-160000, 0, 0)
L7L8	L7	L8	Q235	90.00	4.00	0.0	140000	(-140000, 0, 0)	(140000, 0, 0)
U7U8	U7	U8	Q235	100.00	6.00	0.0	-106667	(106667, 0, 0)	(-106667, 0, 0)
L8L9	L8	L9	Q235	90.00	4.00	0.0	60000	(-60000, 0, 0)	(60000, 0, 0)
U8U9	U8	U9	Q235	100.00	6.00	0.0	-106667	(106667, 0, 0)	(-106667, 0, 0)
L9L10	L9	L10	Q235	90.00	4.00	0.0	60000	(-60000, 0, 0)	(60000, 0, 0)
U9U10	U9	U10	Q235	100.00	6.00	0.0	0	(0, 0, 0)	(0, 0, 0)
L0U0	L0	U0	Q235	70.00	3.00	90.0	-10000	(0, 0, 10000)	(0, 0, -10000)
L1U1	L1	U1	Q235	70.00	3.00	90.0	0	(0, 0, 0)	(0, 0, 0)
L2U2	L2	U2	Q235	70.00	3.00	90.0	-20000	(0, 0, 20000)	(0, 0, -20000)

图 5.13 桁架模型的部分构件信

构件验算结果

	构件名称	状态	轴力(N)	强度验算(MPa)	稳定验算(MPa)	长细比
1	L0L1	通过	60000.0	55.52/241.50	N/A	32.85/350
2	U0U1	通过	0.0	0.00/241.50	N/A	30.03/350
3	L1L2	通过	60000.0	55.52/241.50	N/A	32.85/350
4	U1U2	通过	106666.7	60.20/235.00	60.20 (φ=0.936)	30.03/150
5	L2L3	通过	140000.0	129.54/241.50	N/A	32.85/350
6	U2U3	通过	106666.7	60.20/235.00	60.20 (φ=0.936)	30.03/150
7	L3L4	通过	140000.0	129.54/241.50	N/A	32.85/350
8	U3U4	通过	160000.0	90.30/235.00	90.30 (φ=0.936)	30.03/150
9	L4L5	通过	166666.7	154.22/241.50	N/A	32.85/350
10	U4U5	通过	160000.0	90.30/235.00	90.30 (φ=0.936)	30.03/150
11	L5L6	通过	166666.7	154.22/241.50	N/A	32.85/350
12	U5U6	通过	160000.0	90.30/235.00	90.30 (φ=0.936)	30.03/150
13	L6L7	通过	140000.0	129.54/241.50	N/A	32.85/350
14	U6U7	通过	160000.0	90.30/235.00	90.30 (φ=0.936)	30.03/150
15	L7L8	通过	140000.0	129.54/241.50	N/A	32.85/350
16	U7U8	通过	106666.7	60.20/235.00	60.20 (φ=0.936)	30.03/150
17	L8L9	通过	60000.0	55.52/241.50	N/A	32.85/350
18	U8U9	通过	106666.7	60.20/235.00	60.20 (φ=0.936)	30.03/150
19	L9L10	通过	60000.0	55.52/241.50	N/A	32.85/350
20	U9U10	通过	0.0	0.00/241.50	N/A	30.03/350
21	L0U0	通过	10000.0	15.84/235.00	15.84 (φ=0.790)	63.26/150
22	L1U1	通过	0.0	0.00/241.50	N/A	63.26/350
23	L2U2	通过	20000.0	31.67/235.00	31.67 (φ=0.790)	63.26/150
24	L3U3	通过	0.0	0.00/241.50	N/A	63.26/350
25	L4U4	通过	20000.0	31.67/235.00	31.67 (φ=0.790)	63.26/150

构件验算结果

	构件名称	状态	轴力(N)	强度验算(MPa)	稳定验算(MPa)	长细比
17	L8L9	通过	60000.0	55.52/241.50	N/A	32.85/350
18	U8U9	通过	106666.7	60.20/235.00	60.20 (φ=0.936)	30.03/150
19	L9L10	通过	60000.0	55.52/241.50	N/A	32.85/350
20	U9U10	通过	0.0	0.00/241.50	N/A	30.03/350
21	L0U0	通过	10000.0	15.84/235.00	15.84 (φ=0.790)	63.26/150
22	L1U1	通过	0.0	0.00/241.50	N/A	63.26/350
23	L2U2	通过	20000.0	31.67/235.00	31.67 (φ=0.790)	63.26/150
24	L3U3	通过	0.0	0.00/241.50	N/A	63.26/350
25	L4U4	通过	20000.0	31.67/235.00	31.67 (φ=0.790)	63.26/150
26	L5U5	通过	0.0	0.00/241.50	N/A	63.26/350
27	L6U6	通过	20000.0	31.67/235.00	31.67 (φ=0.790)	63.26/150
28	L7U7	通过	0.0	0.00/241.50	N/A	63.26/350
29	L8U8	通过	20000.0	31.67/235.00	31.67 (φ=0.790)	63.26/150
30	L9U9	通过	0.0	0.00/241.50	N/A	63.26/350
31	L10U10	通过	10000.0	15.84/235.00	15.84 (φ=0.790)	63.26/150
32	L0U1	不通过	108166.5	171.30/235.00	171.30 (φ=0.713)	76.03/150
33	U1L2	通过	84129.5	133.23/241.50	N/A	76.03/350
34	L2U3	通过	60092.5	95.16/235.00	95.16 (φ=0.713)	76.03/150
35	U3L4	通过	36055.5	57.10/241.50	N/A	76.03/350
36	L4U5	通过	12018.5	19.03/235.00	19.03 (φ=0.713)	76.03/150
37	U5L6	通过	12018.5	19.03/235.00	19.03 (φ=0.713)	76.03/150
38	L6U7	通过	36055.5	57.10/241.50	N/A	76.03/350
39	U7L8	通过	60092.5	95.16/235.00	95.16 (φ=0.713)	76.03/150
40	L8U9	通过	84129.5	133.23/241.50	N/A	76.03/350
41	U9L10	不通过	108166.5	171.30/235.00	171.30 (φ=0.713)	76.03/150

图 5.14 桁架模型的构件验算结果

结果显示，除了 L0U1 和 U9L10，其余构件均通过了强度、刚度和稳定性验算。

选取构件 L0U1 进行手算，以检验程序的验算结果。

L0U1 为斜腹杆，轴心受压，其轴力值 $N=-108.17kN$，计算长度 $l_0=l=1803mm$，截面 $\phi 70\times 3$，截面几何特性：$A=6.31cm^2$，$i_x=i_y=2.37cm$。

1. 刚度验算

$$\lambda_x=\lambda_y=\frac{l_0}{i}=\frac{180}{2.37}=75.95<[\lambda]=150$$，满足

2. 整体稳定性验算

按 b 类截面查得稳定系数 $\varphi=0.713$，则：

$$\frac{N}{\varphi A}=\frac{108.17\times 10^3}{0.713\times 6.31\times 10^2}=240.43N/mm^2>235N/mm^2$$，不满足。

手算结果与程序获得的结果一致。

6 总结与展望

6.1. 总结

针对钢结构的 BIM 正向设计需求，本研究开发了平面管桁架参数化 BIM 建模及结构设计程序，在 FreeCAD 和 SAP2000 中分别实现平面管桁架 BIM 模型和有限元模型的参数化建模，在 BIM 模型中基于自然语言荷载指令进行荷载可视化，并通过表格输出桁架结构的构件验算结果，实现了平面管桁架从建模、荷载施加到结构验算的全流程数字化集成。本研究得出以下结论：

1. 参数化建模程序实现了平面管桁架的 BIM 模型的参数化生成，并且通过观察者机制实现模型的动态更新，利用 COM 实时同步 FreeCAD 与 SAP2000，模型同步的响应时间短。这一程序提升了建模速度，特别是在模型体量大或需反复修改时，可以大大缩短设计人员的建模时间。

2. 智能荷载工具集成自然语言处理技术，创新性的开发了基于中文指令的荷载交互系统，支持解析自然语言荷载指令，并在 BIM 模型中通过三维箭头对荷载进行可视化处理，同步加载到有限元模型中。

3. 基于《钢结构设计标准》GB50017-2017，开发构件强度、刚度、稳定性自动化验算程序，解决了 SAP2000 作为力学分析软件无法对结构进行验算的问题。

6.2. 展望

本研究基于平面管桁架结构体系，在钢结构的 BIM 正向设计上做了一定的研究，但为真正实现 BIM 在钢结构设计中的深度应用，还有许多工作尚需完成。

1. 本程序的模型适用范围较窄，目前仅支持平行弦体系的平面管桁架。未来可根据不同结构体系新增设计参数，并且开发空间管桁架参数化建模算法，增加曲面构件拓扑生成功能，适应复杂空间结构需求。

2. 智能荷载工具暂不支持荷载组合，荷载类型的管理系统不完善。虽然本程序区分了恒载和活载两种荷载类型，但并未对其进行组合，可接入 SAP2000 中的荷载组合功能，开发基于自然语言处理的荷载组合功能。

3. 现有验算程序暂未覆盖对节点的设计，节点的验算过程较为繁琐，自动化水平不高。后续可开发节点设计模块，可视化节点的深化设计。

4. 程序对结构的验算功能已基本实现，但设计模块还需完善。程序采用“管道-过滤器”架构，数据流是直线型的，并未真正实现钢结构设计流程中“修改截面-验算-修改截面”这一反复过程。

参考文献

[1] 杨滨赫,周庆旭,吴金钛,周丽娜,程栋.BIM 在新型建筑工业化全过程应用与展望[J].四川建筑,2023,43(6):30-32.

[2] 阎秋.建筑业“十四五”规划出台:到 2035 年,我国全面实现建筑工业化[J].中华建设,2022,(02):12-15.

[3] Borkowski A S. A Literature Review of BIM Definitions: Narrow and Broad Views[J].Technologies 2023,11(6):176.

[4] 杨瑞橦.BIM 技术在建筑智能化建造中的应用[J].居舍,2024,(31):40-43.

[5] 郑杰,陈文强.BIM 在建筑行业推广应用中阻碍因素及对策分析*[J].山西建筑,2020,46(21):185-187.

[6] 叶安杰,陶斯亮,朱伟,等.基于 Revit 的结构 BIM 正向设计技术要点研究[J].浙江建筑,2023,40(2):33-37.

[7] Ruchit Parekh, Dario Trabucco. Recent progress in integrating BIM and LCA for sustainable construction: A Review[J]. International Journal of Science and Research Archive, 2024, 13(1): 907-932.

[8] 陈志翔.大跨度钢桁架结构的刚度与稳定性分析[J].城市建设理论研究(电子版),2024,(18).

[9] 陈振明,王朋,肖运通,等.建筑钢结构智能制造研究及进展[J].建筑钢结构进展,2025,27(1):15-23.

[10] 张易莉.关于超大跨度空间钢结构设计的研究[J].中文科技期刊数据库(文摘版)工程技术,2023,(6):38-41.

[11] 陈光.城市钢桁架过街天桥设计要点[J].科学技术创新,2024,(21):147-150.

[12] 魏勇.南通美术馆复杂钢结构设计[J].建筑结构,2024,54(18): 61-69.

[13] 代春娟.建筑工程管理中 BIM 技术的应用[J].城市建设理论研究(电子版),2025,(3):34-36.

[14] 陈柳花.超大跨度钢桁架桥设计阶段 BIM 技术应用实例[J].城市道桥与防洪,2019,(7):291-295，32.

[15] 周勇,屈闫庆.空间钢结构网架参数化建模与力学仿真分析及研究[J].铁道建筑技术,2018,(8):32-35，59.

[16] Lai Y, Ke K, Wang L, 等. BIM-based intelligent optimization of complex steel joints using SVM and NSGA-II[J]. Journal of Constructional Steel Research, 2024, 223: 109086.

[17] Yao Y, Duan J, Xu K, 等. A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly[J]. High-Confidence Computing, 2024, 4(2): 100211.

[18] Reflections on DeepSeek’s breakthrough[J]. National Science Review, 2025, 12(3): nwaf044.

致 谢

“学科交叉”之风盛行，土木这么大的天地，任何学科都可以和它碰碰面。在土木工程“创新班”中学习，的确是学了很多不怎么“土木”的东西，从《Python》到《机器学习》，学的虽浅，但当我被领着踏入这扇门，就不禁继续向前行。

前行的道路是漫长的，即将攻读硕士学位的我明白我未来的道路还很长。土木是个大天地，很多人都看衰它，但我由衷的不这么认为，它教会我的不只是几门力学和几本规范，更是一种踏实、钻研、创造的精神，希望我能携带这些土木赠与我的，在漫长的未来路上不断求索。

前行的道路是迷茫的，所幸路上不乏领路人。学院的老师具有各种各样的背景，他们极大的开拓了我的视野。陈贤川老师便是一名领路人，他教授的《结构力学》和《钢结构设计原理》课程细致入微、深入原理，又结合紧跟时代、融合最新技术，他开发的程序极具创新性，实用且有趣。他引领我从几乎零基础到开发出自己的程序，在这条路上，我从他身上学习了许多。

前行的道路是乐趣的，时间可以花在热爱之事上，是一件十分幸福的事情。除了前方的领路人，后面还有默默支持和守护着我的家人朋友，他们的守护让我可以全身心投入到我所热爱的事情上。已经不记得有多少次我站在崩溃、失落的边缘，他们耐心劝导，将我拥入温暖的怀中。

学科交叉，我可能即将走往另一条岔路，但我相信有缘自会相见。再见了我的本科四年。

Development of Parametric BIM Modeling and Design System for Planar Tubular Trusses based on FreeCAD and SAP2000

College of Civil Engineering, Shenzhen University

Name: Lin Kaijie Student Number: 2021090102

Instructor: Chen Xianchuan

【**Abstract**】: To address the current challenges of software interoperability barriers and insufficient automation in BIM-based steel structure forward design, this study develops a parametric BIM collaborative design system for planar tubular trusses on the FreeCAD platform. By constructing a three-tier program module based on the "Pipe-Filter" architecture, the research innovatively achieves end-to-end integration of "geometric modeling - load identification - structural verification" for tubular trusses. Key contributions include: A parametric modeling program developed in FreeCAD to generate 3D BIM models of planar tubular trusses, synchronized in real-time with SAP2000; An intelligent load-generation tool integrated with the large language model (DeepSeek), enabling natural language instructions to be converted into structured load parameters, with visualization via 3D arrow symbols; A three-tier verification system compliant with the Chinese code GB50017-2017, utilizing COM interfaces to retrieve SAP2000 finite element analysis results and automate member safety checks. The outcomes provide an open-source solution for intelligent steel structure design, advancing the in-depth application of BIM technology in the design phase.

【**Keywords**】: Building Information Modeling; Parametric design; Natural language processing; Steel structure trusses; FreeCAD

附录：关键代码

1. 参数化 BIM 模型及 SAP2000 有限元模型同步生成

```
    def onChanged(self, obj, prop):
        if prop in ["Span", "Height", "Panels", "TopChordDiameter", "TopChordThickness",
                    "BottomChordDiameter", "BottomChordThickness",
"DiagonalDiameter",
                    "DiagonalThickness", "VerticalDiameter", "VerticalThickness",
"AlternateDiagonals"]:
            self.execute(obj)
            self.update_sap2000(obj)


    def execute(self, obj):
        try:
            doc = obj.Document
            span = obj.Span.Value
            height = obj.Height.Value
            panels = obj.Panels
            # 生成节点坐标
            nodes = self.create_nodes(span, height, panels)
            # 清理旧节点
            if obj.Nodes:
                for node in obj.Nodes:
                    doc.removeObject(node.Name)
                obj.Nodes = []
            # 创建可见节点对象
            new_nodes = []
            for name, pos in nodes.items():
                node_obj = doc.addObject("Part::Vertex", "TrussNode")
                node_obj.Label = name
                node_obj.X = pos.x
```

```
                    node_obj.Y = pos.y
                    node_obj.Z = pos.z
                    node_obj.ViewObject.PointSize = 5
                    node_obj.ViewObject.PointColor = (1.0, 0.0, 0.0)
                    new_nodes.append(node_obj)
                obj.Nodes = new_nodes
                # 创建杆件
                members = list(self.create_members(nodes, panels, obj.AlternateDiagonals,
obj).values())
                obj.Shape = Part.Compound(members)
            except Exception as e:
                App.Console.PrintError(f"执行错误: {str(e)}\n")

    def create_nodes(self, span, height, panels):
        nodes = {}
        dx = span / panels
        # 下弦节点
        for i in range(panels + 1):
            nodes[f"L{i}"] = Base.Vector(i * dx, 0, 0)
        # 上弦节点
        for i in range(panels + 1):
            nodes[f"U{i}"] = Base.Vector(i * dx, 0, height)
        return nodes

    def create_members(self, nodes, panels, alternate, obj):
        members = {}
        # 上下弦杆
        for i in range(panels):
            # 下弦
            members[(f"L{i}", f"L{i + 1}")] = self.create_tube(
```

```
                nodes[f"L{i}"], nodes[f"L{i + 1}"],
                obj.BottomChordDiameter.Value, obj.BottomChordThickness.Value)
            # 上弦
            members[(f"U{i}", f"U{i + 1}")] = self.create_tube(
                nodes[f"U{i}"], nodes[f"U{i + 1}"],
                obj.TopChordDiameter.Value, obj.TopChordThickness.Value)
        # 竖腹杆
        for i in range(panels + 1):
            members[(f"L{i}", f"U{i}")] = self.create_tube(
                nodes[f"L{i}"], nodes[f"U{i}"],
                obj.VerticalDiameter.Value, obj.VerticalThickness.Value)
        # 斜腹杆
        for i in range(panels):
            if alternate and i % 2 == 1:
                members[(f"U{i}", f"L{i + 1}")] = self.create_tube(
                    nodes[f"U{i}"], nodes[f"L{i + 1}"],
                    obj.DiagonalDiameter.Value, obj.DiagonalThickness.Value)
            else:
                members[(f"L{i}", f"U{i + 1}")] = self.create_tube(
                    nodes[f"L{i}"], nodes[f"U{i + 1}"],
                    obj.DiagonalDiameter.Value, obj.DiagonalThickness.Value)
        return members


    def create_tube(self, p1, p2, diameter, thickness):
        direction = p2 - p1
        length = direction.Length
        direction.normalize()
        outer = Part.makeCylinder(diameter / 2, length, p1, direction)
        inner = Part.makeCylinder((diameter / 2) - thickness, length, p1, direction)
        return outer.cut(inner)
```

```
def connect_to_sap2000(self):
    """连接或重新连接到 SAP2000。"""
    try:
        if not self.is_sap_connected:
            self.sap2000 = win32com.client.Dispatch('CSI.SAP2000.API.SapObject')
            self.sap2000.ApplicationStart()
            self.sap_model = self.sap2000.SapModel
            self.is_sap_connected = True
            App.Console.PrintMessage("SAP2000 连接已建立\n")
    except Exception as e:
        App.Console.PrintError(f"连接失败: {str(e)}\n")
        self.is_sap_connected = False


def update_sap2000(self, obj):
    """将更新后的模型推送到 SAP2000。"""
    try:
        nodes = self.create_nodes(obj.Span.Value, obj.Height.Value, obj.Panels)
        members = self.create_members(nodes, obj.Panels, obj.AlternateDiagonals, obj)
        self.update_sap2000_model(obj, nodes, members)
    except Exception as e:
        App.Console.PrintError(f"更新失败: {str(e)}\n")


def update_sap2000_model(self, obj, nodes, members):
    """调用 SAP2000 API 更新模型。"""
    try:
        self.connect_to_sap2000()
        if not self.is_sap_connected:
            raise ConnectionError("无法连接到 SAP2000")
        if not self.is_model_initialized:
            self.sap_model.InitializeNewModel()
```

```
                self.sap_model.SetPresentUnits(9)
                self.is_model_initialized = True
                self.sap_model.File.NewBlank()
            else:
                self.sap_model.File.NewBlank()
            for node_name, node_pos in nodes.items():
                self.sap_model.PointObj.AddCartesian(node_pos.x, node_pos.y, node_pos.z,
node_name, node_name)
            for node, member in members.items():
                start_node = node[0]
                end_node = node[1]
                self.sap_model.FrameObj.AddByPoint(start_node, end_node, "1", "1",
start_node+end_node)
            self.sap_model.PropMaterial.SetMaterial("Q235", 1)
            matName = "Q235"
            fy = 235
            fu = 345
            eFy = 235
            eFu = 345
            SSType = 1
            SSHysType = 2
            StrainAtHardening = 0.002
            StrainAtMaxStress = 0.02
            StrainAtRupture = 0.15
            FinalSlope = -0.1
            self.sap_model.PropMaterial.SetOSteel(matName, fy, fu, eFy, eFu, SSType,
SSHysType,
                                                  StrainAtHardening, StrainAtMaxStress,
StrainAtRupture, FinalSlope)
            def set_pipe(name, d, t):
                if self.sap_model.PropFrame.SetPipe(name, "Q235", d, t) != 0:
```

```
                App.Console.PrintError(f"截面 {name} 设置失败\n")
            set_pipe("TopChord", obj.TopChordDiameter.Value, obj.TopChordThickness.Value)
            set_pipe("BottomChord", obj.BottomChordDiameter.Value,
obj.BottomChordThickness.Value)
            set_pipe("Diagonal", obj.DiagonalDiameter.Value, obj.DiagonalThickness.Value)
            set_pipe("Vertical", obj.VerticalDiameter.Value, obj.VerticalThickness.Value)
            for member_key in members.keys():
                start_node, end_node = member_key
                frame_name = start_node + end_node
                if start_node.startswith("U") and end_node.startswith("U"):
                    self.sap_model.FrameObj.SetSection(frame_name, "TopChord")
                elif start_node.startswith("L") and end_node.startswith("L"):
                    self.sap_model.FrameObj.SetSection(frame_name, "BottomChord")
                elif (start_node.startswith("L") and end_node.startswith("U")) and
start_node[1:] != end_node[1:] or (
                        start_node.startswith("U") and end_node.startswith("L")) and
start_node[1:] != end_node[1:] :
                    self.sap_model.FrameObj.SetSection(frame_name, "Diagonal")
                else:
                    self.sap_model.FrameObj.SetSection(frame_name, "Vertical")
                # 设置 M2、M3 端部释放（两端铰接）
                start_release = [False, False, False, False, True, True]  # U1,U2,U3,R1,R2,R3
                end_release = [False, False, False, False, True, True]
                start_value = [0.0] * 6  # 释放系数，全 0 表示完全释放
                end_value = [0.0] * 6
                ret = self.sap_model.FrameObj.SetReleases(frame_name, start_release,
end_release, start_value, end_value, 0)
                self.sap_model.Analyze.RunAnalysis()

        except Exception as e:
            App.Console.PrintError(f"SAP2000 操作失败: {str(e)}\n")
```

```
            self.is_sap_connected = False
```

2. 智能荷载工具中自然语言处理模块

```
class LoadInstructionParser:
    API_ENDPOINT = "https://api.deepseek.com/v1/chat/completions"
    DIRECTION_MAP = {
        "竖直向下": (0, 0, -1),
        "竖直向上": (0, 0, 1),
        "水平向右": (1, 0, 0),
        "水平向左": (-1, 0, 0),
        "横向": (0, 1, 0),
        "纵向": (1, 0, 0)
    }

    def __init__(self, api_key):
        self.api_key = api_key

    def parse(self, text):
        """
        带重试机制的解析方法
        :param text: 用户输入的指令文本
        :return: 结构化荷载数据（字典）或 None
        """
        for attempt in range(MAX_RETRY):
            try:
                response = self._call_api(text)
                return self._format_response(response)
            except requests.exceptions.RequestException as e:
                if attempt == MAX_RETRY - 1:
                    FreeCAD.Console.PrintError(f"API 请求失败: {str(e)}")
```

```
                return None
            time.sleep(1)

    def _call_api(self, text):
        """安全调用 API 方法"""
        headers = {
            "Authorization": f"Bearer {self.api_key}",
            "Content-Type": "application/json"
        }
        system_prompt = """作为结构工程专家，请从以下指令提取荷载参数，按 JSON 格式返回：
        {
            "load_type": "恒/活",
            "position": {"type": "all/indexes/range/condition", "value": [...]},
            "magnitude": 数值（kN）,
            "direction": "方向描述"
        }"""
        return requests.post(
            self.API_ENDPOINT,
            headers=headers,
            json={
                "model": "deepseek-chat",
                "messages": [
                    {"role": "system", "content": system_prompt},
                    {"role": "user", "content": text}
                ],
                "response_format": {"type": "json_object"},
                "temperature": 0.1
            },
            timeout=100  # 增加超时控制
```

```
        )


    def _format_response(self, response):
        """响应数据格式化和验证"""
        data = json.loads(response.json()['choices'][0]['message']['content'])
        # 数据完整性校验
        required_fields = ['load_type', 'position', 'magnitude', 'direction']
        if not all(field in data for field in required_fields):
            raise ValueError("API 响应缺少必要字段")
        # 荷载类型验证
        if data['load_type'] not in ['恒', '活']:
            raise ValueError(f"无效荷载类型: {data['load_type']}")
        # 数值转换和范围检查
        try:
            data['magnitude'] = float(data['magnitude'])
            if data['magnitude'] <= 0:
                raise ValueError("荷载数值必须大于 0")
        except (TypeError, ValueError):
            raise ValueError("无效荷载数值")
        # 方向向量转换
        direction = data['direction']
        if direction not in self.DIRECTION_MAP:
            raise ValueError(f"未知方向描述: {direction}")
        data['direction_vector'] = self.DIRECTION_MAP[direction]
        return data
```

3. 智能荷载工具中三维可视化模块

```
class LoadApplicator:
    """荷载施加与可视化引擎"""
```

```
def __init__(self, doc, scale_factor=50.0):
        self.doc = doc
        self.scale_factor = scale_factor    # 可视化比例  1kN=50mm
        self.base_radius = 5
        self.COLOR_SCHEME = {
'恒': (1.0, 0.0, 0.0),    # 红色
'活': (0.0, 0.0, 1.0)     # 蓝色
}

def apply(self, load_data):
        """主施加方法"""
        nodes = self._select_nodes(load_data['position'])
        if not nodes:
            raise ValueError("未找到符合条件的节点")
        for node in nodes:
            self._create_arrow(node, load_data)
            self.doc.recompute()
            self._sync_to_sap2000(node, load_data)

def _sync_to_sap2000(self, node, load_data):
        """与 SAP2000 同步荷载"""
        try:
            # 获取连接
            sap_conn = SAP2000Connector()
            if not sap_conn._validate_connection():   # 验证连接是否有效
                sap_conn.reconnect()
            model = sap_conn.sap_model
            if not model:
                raise RuntimeError("SAP2000 连接未建立")
            if model.GetModelIsLocked():
```

```
            raise RuntimeError("模型处于锁定状态，请先保存现有模型")
        # 验证荷载类型映射
        load_info = LOAD_TYPE_MAP.get(load_data['load_type'])
        if not load_info:
            raise ValueError(f"无效荷载类型: {load_data['load_type']}")
        load_name, load_type = load_info
        # 创建荷载模式
        ret, _, pattern_names = model.LoadPatterns.GetNameList()
        if ret != 0:
            raise Exception(f"获取荷载模式失败，错误码：{ret}")
        if load_name not in pattern_names:
            FreeCAD.Console.PrintLog(f"正在创建荷载模式：{load_name}\n")
            if model.LoadPatterns.Add(load_name, load_type, 0, True) != 0:
                raise Exception("创建荷载模式失败")
        # 坐标匹配
                                sap_node = find_matching_node(sap_conn.sap_model,
node.Shape.Vertexes[0].Point)
        if not sap_node:
                    raise ValueError(f"节点匹配失败 | FreeCAD 坐标：
{node.Shape.Vertexes[0].Point}")
        # 转换荷载向量
        direction = load_data['direction_vector']
        if not isinstance(direction, (list, tuple)) or len(direction) != 3:
            raise TypeError("方向向量格式错误，应为三元组")
        force = [load_data['magnitude'] * x for x in direction]
        ret = sap_conn.sap_model.PointObj.SetLoadForce(
            sap_node,          # 节点名称
            load_name,         # 荷载模式
            [force[0], force[1], force[2], 0.0, 0.0, 0.0],  # 6 个元素的数组
            True,              # 覆盖现有荷载
```

```
                "Global",           # 坐标系
                0                   # 相对位置
            )
            # 自动保存
            current_file = sap_conn.sap_model.GetModelFilename()
            save_path = str(current_file)
            ret = sap_conn.sap_model.File.Save(save_path)
            if ret != 0:
                raise Exception(f"自动保存失败，错误码：{ret}")
            FreeCAD.Console.PrintWarning(f"模型已保存至：{save_path}\n")

            FreeCAD.Console.PrintMessage("荷载同步成功！\n")

        except Exception as e:
            error_msg = f"SAP2000 同步失败：{str(e)}"
            FreeCAD.Console.PrintError(error_msg + "\n")
            if 'ret' in locals():
                FreeCAD.Console.PrintError(f"详细错误码：{ret}\n")
            raise RuntimeError(error_msg) from e

    def _load_pattern_exists(self, sap_model, pattern_name):
        """荷载模式检查"""
        ret, num_patterns, pattern_names = sap_model.LoadPatterns.GetNameList()
        if ret != 0:
            raise Exception(f"获取荷载模式失败，错误码：{ret}")
        return pattern_name in pattern_names  # 正确访问名称列表

    def _select_nodes(self, position_info):
        """节点选择路由"""
        all_nodes = TopChordDetector.detect(self.doc)
```

```
        selector_type = position_info['type']

        if selector_type == 'all':
            return all_nodes
        elif selector_type == 'indexes':
            return self._select_by_index(all_nodes, position_info['value'])
        elif selector_type == 'condition':
            return self._filter_by_condition(all_nodes, position_info['value'])
        else:
            return []

    def _select_by_index(self, nodes, indexes):
        """索引选择器（1-based 转 0-based）"""
        valid_indexes = [i-1 for i in indexes if isinstance(i, int) and i > 0]
        return [nodes[i] for i in valid_indexes if i < len(nodes)]

    def _filter_by_condition(self, nodes, condition):
        """安全条件筛选器"""
        filtered = []
        for node in nodes:
            point = node.Shape.Vertexes[0].Point
            try:
                # 使用限制环境执行 eval
                if eval(condition, {'X': point.x, 'Y': point.y, 'Z': point.z}, {}):
                    filtered.append(node)
            except:
                continue
        return filtered
```

```
def _create_arrow(self, node, load_data):
    """参数化创建荷载箭头"""
    pos = node.Shape.Vertexes[0].Point
    direction = load_data['direction_vector']
    length = load_data['magnitude'] * self.scale_factor          #1kN 对应 50mm

    # 创建箭头组件
    shaft = self._create_shaft(pos, direction, length)
    head = self._create_head(pos, direction, length)

    # 组合对象并设置颜色
    if str(load_data['load_type']) == '恒':
        type = 'dead'
    else:
        type = 'live'
    load_name = type + str(load_data['magnitude'])
    compound = self.doc.addObject("Part::Compound", load_name)
    compound.Links = [shaft, head]
    compound.ViewObject.ShapeColor = self.COLOR_SCHEME[load_data['load_type']]
    del shaft, head

    # 注册到管理器
    ArrowScaleManager().register_arrow(
        compound,
        load_data['magnitude'],
        direction,
        pos
    )
    return compound
```

```
    def _create_shaft(self, pos, direction, length):
        """创建箭头杆部"""
        shaft = self.doc.addObject("Part::Cylinder", "LoadShaft")
        shaft.Radius = self.base_radius * self.scale_factor
        shaft.Height = length
        shaft.Placement = self._create_placement(pos, direction, offset=-(1.2*length+50))
        return shaft

    def _create_head(self, pos, direction, length):
        """创建箭头头部"""
        head = self.doc.addObject("Part::Cone", "LoadHead")
        head.Radius1 = self.base_radius * 3 * self.scale_factor
        head.Radius2 = 0
        head.Height = length * 0.2
        head.Placement = self._create_placement(
            pos, direction,    offset = -(0.2*length+50)
        )
        return head

    @staticmethod
    def _create_placement(base_pos, direction, offset=0, rotation_axis=FreeCAD.Vector(0,0,1)):
        """创建三维空间变换"""
        return FreeCAD.Placement(
            base_pos + FreeCAD.Vector(*direction) * offset,
             FreeCAD.Rotation(rotation_axis, FreeCAD.Vector(*direction))
        )
```

4. SAP2000 连接器

```
class SAP2000Connector:
    """SAP2000 连接器"""
```

```
_instance = None
_is_connected = False
_lock = threading.Lock()  # 线程安全锁

def __new__(cls):
    with cls._lock:
        if not cls._instance:
            cls._instance = super().__new__(cls)
            # 初始化成员变量
            cls._instance._sap2000 = None
            cls._instance._sap_model = None
            cls._instance.model_path = None
            cls._instance._connection_attempts = 0
        return cls._instance

@property
def sap_model(self):
    return self._sap_model

def set_model_path(self, path):
    """设置模型路径"""
    if not path:
        return False
    path = os.path.normpath(path)
    if not os.path.isfile(path):
        raise FileNotFoundError(f"模型文件不存在: {path}")
    if not path.lower().endswith(('.sdb', '.s2k')):
        raise ValueError("仅支持 .sdb 和 .s2k 格式")

    with self._lock:
```

```
            if self.model_path != path:
                self.model_path = path
                self._cleanup_connection()
                FreeCAD.Console.PrintLog(f"模型路径更新: {path}\n")

        return True

    def _validate_connection(self):
        """深度连接验证"""
        try:
            if not self._sap_model:
                return False
            # 通过获取基础信息验证连接
            ret, filename = self._sap_model.GetModelFilename()
            return ret == 0 and filename == self.model_path
        except:
            return False

    def _init_connection(self):
        """核心连接逻辑"""
        try:
            if self._is_connected:
                return True
            if not self.model_path:
                raise RuntimeError("未选择模型文件")
            self._sap2000 = win32com.client.Dispatch('CSI.SAP2000.API.SapObject')
            self._sap2000.ApplicationStart()
            self._sap_model = self._sap2000.SapModel
            # 打开模型文件
            ret = self._sap_model.File.OpenFile(self.model_path)
```

```
            if ret != 0:
                raise ConnectionError(f"打开模型失败 (错误码: {ret})")
            # 配置单位制
            current_units = self._sap_model.GetPresentUnits()
            if current_units != 5:  # 5 = kN-mm
                self._sap_model.SetPresentUnits(5)
                FreeCAD.Console.PrintWarning("单位制已自动转换为 kN-mm\n")
            # 验证模型状态
            if self._sap_model.GetModelIsLocked():
                raise ConnectionAbortedError("模型已被锁定（请检查 SAP2000 界面）")
            self._is_connected = True
            self._connection_attempts = 0
            FreeCAD.Console.PrintMessage("成功连接 SAP2000 模型\n")
            return True
        except Exception as e:
            self._connection_attempts += 1
            self._cleanup_connection()
            if self._connection_attempts <= 3:
                FreeCAD.Console.PrintWarning(f"连接尝试失败({self._connection_attempts}/3): {str(e)}\n")
            else:
                raise RuntimeError("连接尝试超过最大次数") from e

    def select_model_interactive(self):
        """交互式文件选择"""
        from PySide2 import QtWidgets
        try:
            path, _ = QtWidgets.QFileDialog.getOpenFileName(
                None,
                "选择 SAP2000 模型文件",
```

```
                self._get_default_path(),
                "SAP2000 Files (*.sdb *.s2k);;All Files (*)"
            )
            if path and self.set_model_path(path):
                FreeCAD.Console.PrintMessage(f"已选择模型: {os.path.basename(path)}\n")
                return True
            return False
        except Exception as e:
            FreeCAD.Console.Pri
```

5. 结构分析工具中构件验算模块

```
class MemberCheck:
    """钢管构件验算"""

    # ----------------------------
    #  嵌套数据类定义
    # ----------------------------
    @dataclass
    class _HollowSection:
        """空心圆管截面参数"""
        D: float    #  外径(mm)
        t: float    #  壁厚(mm)

        def __post_init__(self):
            """数据有效性验证"""
            if self.t >= self.D / 2:
                raise ValueError("壁厚不能超过半径")
            if self.D <= 0 or self.t <= 0:
                raise ValueError("尺寸必须为正数")
```

```
@dataclass
class _SteelMaterial:
    """钢材材料属性"""
    fy: float   # 屈服强度(MPa)
    fu: float   # 抗拉强度(MPa)
    E: float = 206000   # 弹性模量 (MPa)，默认 Q235~Q460 钢

    def __post_init__(self):
        if self.fy <= 0 or self.fu <= 0:
            raise ValueError("强度参数必须为正数")

@dataclass
class _MemberForce:
    """构件内力"""
    N: float   # 轴力 (kN)，压力为负，拉力为正

    def __post_init__(self):
        if abs(self.N) > 1e6:
            raise ValueError("轴力值超出合理范围")

@dataclass
class _MemberGeometry:
    """构件几何参数"""
    L: float   # 几何长度(mm)
    μ: float = 1.0   # 计算长度系数（表 7.4.1）

    def __post_init__(self):
        if self.L <= 0:
            raise ValueError("长度必须为正数")
```

```
# ---------------------------
# 初始化方法
# ---------------------------
def __init__(self,
                 D: float, t: float,
                 fy: float, fu: float,
                 N: float, L: float):
    """
    参数初始化入口[1](@ref)
    :param D: 外径(mm)
    :param t: 壁厚(mm)
    :param fy: 屈服强度(MPa)
    :param fu: 抗拉强度(MPa)
    :param N: 轴力(kN)
    :param L: 几何长度(mm)
    """
    # 实例化内部数据类
    self.section = self._HollowSection(D, t)
    self.material = self._SteelMaterial(fy, fu)
    self.force = self._MemberForce(N)
    self.geometry = self._MemberGeometry(L)

    # 计算截面特性
    self.section_props = self._calculate_section_properties()

    # 存储验算结果
    self.result = {
        'status': "通过",
        'checks': [],
        'N_Ed': abs(N)  # 轴力设计值（N）
```

```
        }


    # ----------------------------
    #  核心计算方法
    # ----------------------------
    def _calculate_section_properties(self) -> dict:
        """计算截面几何特性"""
        D, t = self.section.D, self.section.t
        d = D - 2 * t   #  内径
        A = math.pi * (D ** 2 - d ** 2) / 4   #  截面积  (mm²)
        I = math.pi * (D ** 4 - d ** 4) / 64   #  惯性矩  (mm⁴)
        i = math.sqrt(I / A)   #  回转半径
        return {'A': A, 'I': I, 'i': i}


    def _verify_strength(self):
        """执行强度验算"""
        σ = self.result['N_Ed'] / self.section_props['A']   #  压力强度设计值
        limit = self.material.fy if self.force.N < 0 else 0.7 * self.material.fu   #  拉力
        self._add_check('强度验算', σ, limit)


    def _verify_stability(self):
        """执行稳定性验算"""
        check_data = {
        'name': '稳定验算',
        'value': 'N/A',
        'limit': 'N/A',
        'passed': True,
        'note': '受拉构件不验算稳定性' if self.force.N >=0 else None
    }
        if self.force.N < 0:
```

```
            # 正则化长细比计算
            λ = (self.geometry.μ * self.geometry.L) / self.section_props['i']
            λn = (λ / math.pi) * math.sqrt(self.material.fy / self.material.E)

            # 稳定系数计算（b 类截面参数）可根据后续需求查表更改
            α1, α2, α3 = 0.65, 0.965, 0.300  # 附录 D 表 D.0.1
            φ = (1.0 - α1 * λn ** 2) if λn <= 0.215 else (
                    (α2 + α3 * λn + λn ** 2 - math.sqrt((α2 + α3 * λn + λn ** 2) ** 2 - 4 * λn
** 2)) / (2 * λn ** 2))

            σ_cr = φ * self.material.fy
            self._add_check('稳定验算',
                            self.result['N_Ed'] / self.section_props['A'],
                            σ_cr,
                            extra_data={'φ': φ, 'λn': λn})

    def _verify_slenderness(self):
        """长细比验算"""
        λ = (self.geometry.μ * self.geometry.L) / self.section_props['i']
        limit = 150 if self.force.N < 0 else 350
        self._add_check('长细比', λ, limit)

    def _add_check(self, name, value, limit, extra_data=None):
        """添加验算记录"""
        check = {
            'name': name,
            'value': value,
            'limit': limit,
            'passed': value <= limit
```

```
        }
        if extra_data:
            check.update(extra_data)
        self.result['checks'].append(check)

        # 更新整体状态
        if not check['passed']:
            self.result['status'] = "不通过"

    # ----------------------------
    # 公共接口方法
    # ----------------------------
    def perform_check(self) -> dict:
        """执行完整验算流程"""
        self._verify_strength()
        self._verify_stability()
        self._verify_slenderness()
        return self.result
```

6. 构件验算数据可视化表格

```
class ResultDialog(QtGui.QDialog):
    """显示验算结果的表格对话框"""
    def __init__(self, results, parent=None):
        super().__init__(parent)
        self.results = results
        self._init_ui()

    def _init_ui(self):
        # 安全获取检查项的方法
        def get_check(checks, name):
```

```
        for item in checks:
            if item['name'] == name:
                return item
        return {'value': 'N/A', 'limit': 'N/A'}  # 默认值
self.setWindowTitle("构件验算结果")
self.setMinimumSize(1200, 800)

layout = QtGui.QVBoxLayout()
table = QtGui.QTableWidget()
# 设置表格列
headers = [
    "构件名称", "状态", "轴力(N)",
    "强度验算(MPa)", "稳定验算(MPa)", "长细比"
]
table.setColumnCount(len(headers))
table.setHorizontalHeaderLabels(headers)
# 填充数据
table.setRowCount(len(self.results))
for row, (name, result) in enumerate(self.results):
    table.setItem(row, 0, QtGui.QTableWidgetItem(name))
    table.setItem(row, 1, QtGui.QTableWidgetItem(result['status']))
    table.setItem(row, 2, QtGui.QTableWidgetItem(f"{abs(result['N_Ed']):.1f}"))
    # 提取各检查项结果
    strength = get_check(result['checks'], '强度验算')
    stability = get_check(result['checks'], '稳定验算')
    slenderness = get_check(result['checks'], '长细比')
    # 强度验算列
    strength_value = strength.get('value', 'N/A')
    strength_limit = strength.get('limit', 'N/A')
                    strength_text = f"{strength_value:.2f}/{strength_limit:.2f}" if
```

```
isinstance(strength_value, (float, int)) and isinstance(strength_limit, (float, int)) else f"{strength_value}/{strength_limit}"
                table.setItem(row, 3, QtGui.QTableWidgetItem(strength_text))
                # 稳定验算列
                stability_value = stability.get('value', 'N/A')
                stability_text = f"{stability_value:.2f}" if isinstance(stability_value, (float, int)) else f"{stability_value}"
                if 'φ' in stability:
                    stability_text += f" (φ={stability['φ']:.3f})"
                elif stability.get('note'):
                    stability_text += f" ({stability['note']})"
                table.setItem(row, 4, QtGui.QTableWidgetItem(stability_text))
                # 长细比列
                slenderness_value = slenderness.get('value', 'N/A')
                slenderness_limit = slenderness.get('limit', 'N/A')
                    slenderness_text = f"{slenderness_value:.2f}/{slenderness_limit}" if isinstance(slenderness_value, (float, int)) else f"{slenderness_value}/{slenderness_limit}"
                table.setItem(row, 5, QtGui.QTableWidgetItem(slenderness_text))
            table.resizeColumnsToContents()
            layout.addWidget(table)
            self.setLayout(layout)
```