|  |  |
| --- | --- |
| 文档标识： |  |
| 版本号： | |

**“科梁杯第一题”**

**设计说明**

|  |  |
| --- | --- |
| 编制： | 年 月 日 |
| 审核： | 年 月 日 |
| 批准： | 年 月 日 |

|  |
| --- |
| 长沙科梁科技有限公司 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编写/修改记录** | | | | |
| **序号** | **版本号** | **编写/修改时间** | **修改说明（修改章节及内容概要）** | **修改人** |
|  | V0.1 | 2023.12.09 | 完成首版本 |  |
|  | V0.2 | 2023.12.16 | 去除含有团队的名称以及符号 |  |
|  | V1.0 | 2023.12.17 | 调整格式 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1 范围 1](#_Toc153543887)

[1.1 标识 1](#_Toc153543888)

[1.2 系统概述 1](#_Toc153543889)

[1.2.2 软件用途 1](#_Toc153543890)

[1.2.2 软件概述 1](#_Toc153543891)

[1.3 文档概述 2](#_Toc153543892)

[2 引用文档 3](#_Toc153543893)

[3 CSCI 级设计决策 3](#_Toc153543894)

[4 CSCI体系结构设计 3](#_Toc153543895)

[4.1 CSCI 部件 3](#_Toc153543896)

[4.1.1 软件单元 4](#_Toc153543897)

[4.1.2 模型数据（ARMU1-SDD-MDDAT） 4](#_Toc153543898)

[4.1.3 解析模块（ARMU1-SDD-PRASE） 4](#_Toc153543899)

[4.1.4 矩阵模块（ARMU1-SDD-MATRXI） 20](#_Toc153543900)

[4.1.5 仿真模块（ARMU1-SDD-SIMU） 24](#_Toc153543901)

[4.2 执行方案 30](#_Toc153543902)

[4.3 接口设计 31](#_Toc153543903)

[4.3.1 内部接口 31](#_Toc153543904)

[4.3.2 外部接口 37](#_Toc153543905)

[5 CSCI详细设计 38](#_Toc153543906)

[5.1 仿真模型 38](#_Toc153543907)

[6 需求可追踪性 39](#_Toc153543908)

[7 注解 39](#_Toc153543909)

[附录A XXX要求 40](#_Toc153543910)

XXX软件

设计说明

* 1. 范围
     1. 标识

1. 文档标识号：KLBFQ-SDD；
2. 文档名称：科梁杯第一题软件设计说明；
3. 缩略名：KLBFQ；
4. 版本号和发布号：V1.0。
   * 1. 系统概述
5. 软件用途

“科梁杯”第一题仿真软件（以下简称仿真软件）用于在windows10环境下对含子功能模块 Gain（增益器）、Sum（求和）、Mult（乘法器）、Sine（正弦波）、Cons（常量）、Disp（数据显示）的模型进行数字仿真。用户通过修改模型配置文件更改模型，通过软件对该模型进行仿真，得到仿真计算结果，其系统概述如图 1所示。

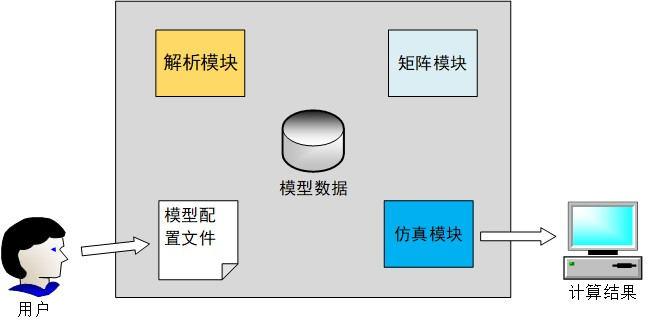


图 1系统概述图

1. 软件概述

仿真软件运行于windows10机器上，用于对用户提供的模型配置文件进行模型仿真。该软件采用模块化设计，模块分为解析模块、矩阵模块、仿真模块以及公共的模型数据，如图 2所示。各模块之间的用途如下：

* 解析模块：对模型文件进行解析，将解析后的模型信息存入到模型数据中。
* 矩阵模块：根据模型中各个模块之间的关系创建零阶矩阵，用于描述模型关系。
* 仿真模块：对模型进行仿真计算。
* 模型数据：保存模型的信息，如输入模块、输出模块名称、模块参数、模块关系和模块列表。

软件通过解析模块解析模型配置文件，并将解析后的模型数据存入到模型数据中，后通过矩阵模块为模型数据中各个模块的关系创建零阶矩阵，最后仿真模块根据零阶矩阵进行深度优先搜索查看是否含有代数环，最后该模块通过后序递归对模型进行仿真计算，并将每一步的仿真结果保存。



图 2 软件结构设计图

* + 1. 文档概述

本文档内容依据长沙科梁科技有限公司《关于举办第一届“科梁杯”程序设计挑战赛的通知》的约束，对每项需求提供实现的方案。

第1节主要概述了文档的标识和内容。

第2节列出了本文档所引用的相关标准规范和技术文档。

第3节列出了本文档针对需求所制定的设计决策。

第4节主要针对系统的体系结构设计及之间的依赖关系。

第5节提供了设计的可追踪性。

第6节提供了本文档常用的关键字及相关注释信息

* 1. 引用文档

引用文档如表 1所示。

表 1 引用文档

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **版本** | **来源** |
| 1 | C++编码规范 | - | V1.2 | 内部 |
| 2 | 关于举办第一届“科梁杯”程序设计挑战赛的通知 | - | - | 内部 |

* 1. CSCI 级设计决策

对仿真软件进行进一步的需求分析后，关键性需求，如表 2所示。

表 2 关键性需求表

|  |  |  |  |
| --- | --- | --- | --- |
| 需求 | 功能 | 质量属性 | 约束 |
| 业务级需求 | 仿真模型 | 仿真结果与matlab仿真一致 | 1、使用windows环境  2、仅限于Gain（增益器）、Sum（求和）、Mult（乘法器）、Sine（正弦波）、Cons（常量）、Disp（数据显示）这5类模块的模型仿真 |
| 用户级需求 | 动态调整子模块执行顺序 | 仿真结果与预期结果一致 |
| 动态调参 |
| 开发级需求 | 支持成环计算 | 仿真结果与matlab仿真一致 |

对上述关键性需求进行分析，得到以下CSCI的设计决策。

（1）对于业务级需求，重点在于模型的仿真计算，在该设计中，我们采用后续递归的方式，从disp模块开始，逐步的往上进行递归求取每一个模块在当前步的值。

（2）对于用户级需求，主要在于子模块的调整和动态调参。子模块的调整方面，采用json文件表示模型的相关信息，其中包含了各个模块之间的关系和模块的参数，可由用户修改该配置文件从而达到用户期望的模型。

（3）对于开发及需求，重点在于模型中含有代数环的计算，当模型中含有环时，将采用类似于simulink中的memory模块，使用上一步的值进行当前步的计算。另外也可以在配置文件中指定其初始值作为第一步的模型计算。

* 1. CSCI体系结构设计
     1. CSCI 部件

#### 软件单元

仿真软件按照模块划分，分为四个模块，分别为解析模块、矩阵模块、仿真模块以及公共的模型数据。各模块之间的职责如表 3所示。

表 3 模块职责

|  |  |  |
| --- | --- | --- |
| 序号 | 模块名称 | 模块职责 |
| 1 | 解析模块 | 解析模型配置文件 |
| 2 | 矩阵模块 | 为模型关系创建零阶矩阵 |
| 3 | 仿真模块 | 对模型进行计算 |
| 4 | 模型数据 | 提供模型数据 |

#### 模型数据（ARMU1-SDD-MDDAT）

模型数据模块作为公共模块，用于存储模型的信息并给其他模块提供模型信息。该模块包含两个文件，分别为modeldata.h和modeldata.cpp。，由于该文件为公共数据文件，不存在任何函数接口，其模型数据详细说明见表 4。

表 4 模型数据模块数据描述

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 数据名称 | 数据定义 | 数据描述 |
| 1 | m\_step | static Timing | 步长信息，包含了仿真时长和仿真步长 |
| 2 | m\_gainModule | static std::map<std::string, Gain> | 所有的Gain模块信息 |
| 3 | m\_sumModule | static std::map<std::string, ModuleInfo> | 所有的Sum模块信息 |
| 4 | m\_multModule | static std::map<std::string, ModuleInfo> | 所有的Mult模块信息 |
| 5 | m\_sineModule | static std::map<std::string, Sine> | 所有的Sine模块信息 |
| 6 | m\_consModule | static std::map<std::string, Cons> | 所有的Cons模块信息 |
| 7 | m\_dispModule | static std::map<std::string, Disp> | 所有的Disp模块信息 |
|  | m\_moduleList | static std::vector<std::string> | 模块列表 |
|  | m\_relationship | static std::vector<std::vector<std::string>> | 模块之间的关系 |

#### 解析模块（ARMU1-SDD-PRASE）

解析模块共包含1个.cpp文件，详细说明见表 5。

表 5 解析模块源文件列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 程序文件 | 概述 | 服务对象 | 函数个数 |
| 1 | parsecfg.cpp | 解析模型配置文件 | 模型数据 | 24 |

##### parsecfg.cpp（ARMU1-SDD-PRASE-001）

parsecfg.cpp的主要用途为解析模型配置文件，该模块的简要描述见表 6，其函数列表见表 7。

表 6 parsecfg.cpp简要描述

|  |  |  |
| --- | --- | --- |
| 源文件名称 | 对外提供接口 | 涉及用例 |
| parsecfg.cpp | bool parseJsonConf() | 仿真模型  动态调整子模块执行顺序  动态调参  支持成环计算 |
| static ParseCfg getInstance() |

表 7 parsecfg.cpp函数列表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 函数定义 | 用途 | 对外接口 |
| 1 | ParseCfg() | 构造函数 | 否 |
| 2 | ~ParseCfg() | 析构函数 | 否 |
| 3 | ParseCfg(const ParseCfg&my\_simulation) = delete; | 禁止外部复制构造 | 否 |
| 4 | const ParseCfg& operator=(const ParseCfg& my\_simulation) = delete; | 禁止外部赋值 | 否 |
| 5 | void parseGainModule  (const rapidjson::Document& doc); | 解析Gain模块 | 否 |
| 6 | void parseSumModule  (const rapidjson::Document& doc); | 解析Sum模块 | 否 |
| 7 | void parseMultModule  (const rapidjson::Document& doc); | 解析Mult模块 | 否 |
| 8 | void parseConsModule  (const rapidjson::Document& doc); | 解析Cons模块 | 否 |
| 9 | void parseSineModule  (const rapidjson::Document& doc); | 解析Sine模块 | 否 |
| 10 | void parseDspModule  (const rapidjson::Document& doc); | 解析Disp模块 | 否 |
| 11 | bool moduleValidityCheck(); | 对模型进行合法性检查 | 否 |
| 12 | bool checkGain(); | 检查Gain模块 | 否 |
| 13 | bool checkSum(); | 检查Sum模块 | 否 |
| 14 | bool checkMult(); | 检查Mult模块 | 否 |
| 15 | bool checkDisp(); | 检查Disp模块 | 否 |
| 16 | bool checkSine(); | 检查Sine模块 | 否 |
| 17 | bool checkStep(); | 检查步长是否合法 | 否 |
| 18 | bool ioCheck  (const std::string\* io\_list, const int& list\_size); | 对模块的输入输出进行合法性检查 | 否 |
| 19 | void parseStepSize  (const rapidjson::Document& doc); | 解析步长信息 | 否 |
| 20 | void recordModuleNumber  (const std::string& name); | 记录模块的名称 | 否 |
| 21 | void recordRelationship  (const std::string& from,  const std::string& to); | 记录模块关系 | 否 |
| 22 | int getModulePosition  (std::string module\_name); | 获取模块ID | 否 |
| 23 | static ParseCfg& getInstance() | 获取单例 | 是 |
| 24 | bool parseJsonConf(); | 解析模型配置文件入口 | 是 |

###### 构造函数（ARMU1-SDD-PRASE-001-001）

ParseCfg()函数见表 8

表 8 ParseCfg()函数

|  |  |
| --- | --- |
| 函数定义 | ParseCfg() |
| 功能描述 | 构造函数 |
| 输入 | 无 |
| 输出 | 无 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

###### 析构函数（ARMU1-SDD-PRASE-001-002）

~ParseCfg()函数见表 9

表 9 ParseCfg()函数

|  |  |
| --- | --- |
| 函数定义 | ParseCfg() |
| 功能描述 | 构造函数 |
| 输入 | 无 |
| 输出 | 无 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

###### 析构函数（ARMU1-SDD-PRASE-001-003）

~ParseCfg()函数见表 10。

表 10 ParseCfg()函数

|  |  |
| --- | --- |
| 函数定义 | ~ParseCfg() |
| 功能描述 | 析构函数 |
| 输入 | 无 |
| 输出 | 无 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

###### 禁止外部复制构造（ARMU1-SDD-PRASE-001-004）

~ParseCfg()函数见表 11。

表 11 ParseCfg(const ParseCfg&my\_simulation)函数

|  |  |
| --- | --- |
| 函数定义 | ParseCfg(const ParseCfg&my\_simulation) = delete; |
| 功能描述 | 禁止外部复制构造 |
| 输入 | const ParseCfg& my\_simulation |
| 输出 | 无 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

###### 禁止外部赋值（ARMU1-SDD-PRASE-001-005）

const ParseCfg& operator=(const ParseCfg& my\_simulation)函数见表 12。

表 12 const ParseCfg& operator=(const ParseCfg& my\_simulation)函数

|  |  |
| --- | --- |
| 函数定义 | const ParseCfg& operator=(const ParseCfg& my\_simulation) = delete; |
| 功能描述 | 禁止外部赋值 |
| 输入 | const ParseCfg& my\_simulation |
| 输出 | 无 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

###### 解析Gain模块（ARMU1-SDD-PRASE-001-006）

parseGainModule函数见表 13。

表 13 parseGainModule函数

|  |  |
| --- | --- |
| 函数定义 | void parseGainModule(const rapidjson::Document& doc); |
| 功能描述 | 解析Gain模块信息 |
| 输入 | const rapidjson::Document& doc：配置文件文本信息 |
| 输出 | std::map<std::string, Gain>m\_gainModule;  std::vector<std::vector<std::string>>m\_relationship;  std::vector<std::string>m\_moduleList; |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、判断doc中是否有gain字段的配置  2、提取gain模块的名称到m\_gainModule中  3、记录gain模块的名称到m\_moduleList中  4、提取gain模块的增益值到m\_gainModule中  5、提取gain模块的输入到m\_gainModule中  6、提取gain模块的输出m\_gainModule中  7、提取gain模块的初始值到m\_gainModule中  8、记录gain和其他模块之间的关系到m\_relationship中  9、记录gain的初值到m\_gainModule |
| 人机交互界面设计 | 无 |

###### 解析Sum模块（ARMU1-SDD-PRASE-001-007）

parseSumModule函数见表 14。

表 14 parseSumModule函数

|  |  |
| --- | --- |
| 函数定义 | void parseSumModule(const rapidjson::Document& doc); |
| 功能描述 | 解析Sum模块信息 |
| 输入 | const rapidjson::Document& doc：配置文件文本信息 |
| 输出 | std::map<std::string, ModuleInfo> ModuleData::m\_sumModule  std::vector<std::vector<std::string>>m\_relationship  std::vector<std::string>m\_moduleList |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、判断doc中是否有sum字段的配置  2、提取sum模块的名称到m\_sumModule中  3、记录sum模块的名称到m\_moduleList中  4、提取sum模块的输入到m\_sumModule中  5、提取sum模块的输出m\_sumModule中  6、提取sum模块的初始值到m\_sumModule中  7、记录sum和其他模块之间的关系到m\_relationship中  8、记录gain的初值到m\_sumModule |
| 人机交互界面设计 | 无 |

###### 解析Mult模块（ARMU1-SDD-PRASE-001-008）

parseMultModule函数见表 15。

表 15 parseMultModule函数

|  |  |
| --- | --- |
| 函数定义 | void parseMultModule(const rapidjson::Document& doc); |
| 功能描述 | 解析Mult模块信息 |
| 输入 | const rapidjson::Document& doc：配置文件文本信息 |
| 输出 | std::map<std::string, ModuleInfo> ModuleData::m\_multModule  std::vector<std::vector<std::string>>m\_relationship  std::vector<std::string>m\_moduleList |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、判断doc中是否有sum字段的配置  2、提取mult模块的名称到m\_multModule中  3、记录mult模块的名称到m\_moduleList中  4、提取mult模块的输入到m\_multModule中  5、提取mult模块的输出m\_multModule中  6、提取mult模块的初始值到m\_multModule中  7、记录mult和其他模块之间的关系到m\_relationship中  8、记录mult的初值到m\_multModule |
| 人机交互界面设计 | 无 |

###### 解析Cons模块（ARMU1-SDD-PRASE-001-009）

parseConsModule函数见表 16。

表 16 parseConsModule函数

|  |  |
| --- | --- |
| 函数定义 | void parseConsModule(const rapidjson::Document& doc) |
| 功能描述 | 解析Cons模块信息 |
| 输入 | const rapidjson::Document& doc：配置文件文本信息 |
| 输出 | std::map<std::string, ModuleInfo>m\_consModule  std::vector<std::vector<std::string>>m\_relationship  std::vector<std::string>m\_moduleList |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、判断doc中是否有sum字段的配置  2、提取cons模块的名称到m\_consModule中  3、记录cons模块的名称到m\_moduleList中  3、记录cons模块的参数值到m\_consModule中  4、提取cons模块的输出m\_consModule中  5、记录cons和其他模块之间的关系到m\_relationship中 |
| 人机交互界面设计 | 无 |

###### 解析Sine模块（ARMU1-SDD-PRASE-001-010）

parseSineModule函数见表 17。

表 17 parseSineModule函数

|  |  |
| --- | --- |
| 函数定义 | void parseSineModule(const rapidjson::Document& doc); |
| 功能描述 | 解析Sine模块信息 |
| 输入 | const rapidjson::Document& doc：配置文件文本信息 |
| 输出 | std::map<std::string, ModuleInfo>m\_sineModule  std::vector<std::vector<std::string>>m\_relationship  std::vector<std::string>m\_moduleList |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、判断doc中是否有sum字段的配置  2、提取sine模块的名称到m\_sineModule中  3、记录sine模块的名称到m\_moduleList中  4、记录sine模块的幅值到m\_sineModule中  5、提取sine模块的输出m\_sineModule中  6、记录sine和其他模块之间的关系到m\_relationship中 |
| 人机交互界面设计 | 无 |

###### 解析Disp模块（ARMU1-SDD-PRASE-001-011）

parseDspModule函数见表 18。

表 18 parseDspModule函数

|  |  |
| --- | --- |
| 函数定义 | void parseDspModule(const rapidjson::Document& doc); |
| 功能描述 | 解析Disp模块信息 |
| 输入 | const rapidjson::Document& doc：配置文件文本信息 |
| 输出 | std::map<std::string, ModuleInfo> m\_dispModule  std::vector<std::vector<std::string>>m\_relationship  std::vector<std::string>m\_moduleList |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、判断doc中是否有sum字段的配置  2、提取Disp模块的名称到m\_dispModule中  3、记录Disp模块的名称到m\_moduleList中  4、提取Disp模块的输入m\_sineModule中  5、记录Disp和其他模块之间的关系到m\_relationship中 |
| 人机交互界面设计 | 无 |

###### 对模型进行合法性检查（ARMU1-SDD-PRASE-001-012）

moduleValidityCheck函数见表 19。

表 19 moduleValidityCheck函数

|  |  |
| --- | --- |
| 函数定义 | bool moduleValidityCheck(); |
| 功能描述 | 对模型进行合法性检查 |
| 输入 | m\_step  m\_gainModule  m\_sumModule  m\_multModule  m\_sineModule  m\_consModule  m\_dispModule  m\_moduleList  m\_relationship |
| 输出 | 无 |
| 返回值 | true：模型合法  false：模型不合法 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 检查gain模块 2. 检查sum模块 3. 检查mult模块 4. 检查disp模块 5. 检查sine模块 6. 检查cons模块 7. 检查步长信息 |
| 人机交互界面设计 | 无 |

###### 检查Gain模块（ARMU1-SDD-PRASE-001-013）

checkGain函数见表 20。

表 20 checkGain函数

|  |  |
| --- | --- |
| 函数定义 | bool checkGain(); |
| 功能描述 | 检查Gian模块是否合法 |
| 输入 | m\_gainModule |
| 输出 | 无 |
| 返回值 | true：模型合法  false：模型不合法 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 遍历gain的所有模块 2. 检查gain模块的输入是否为空，如果为空，则返回false，并输出日志到控制台 3. 检查gain模块的输出是否为空，如果为空，则返回false，并输出日志到控制台。 4. 检查gain的输入模块是否存在，如果不存在，则返回false，并输出日志到控制台 5. 检查gain的输出模块是否存在，如果不存在，则返回false，并输出日志到控制台   6、若检查都通过则返回true |
| 人机交互界面设计 | 无 |

###### 检查Sum模块（ARMU1-SDD-PRASE-001-014）

checkSum函数见表 21。

表 21 checkSum函数

|  |  |
| --- | --- |
| 函数定义 | bool checkSum(); |
| 功能描述 | 检查Sum模块是否合法 |
| 输入 | m\_sumModule |
| 输出 | 无 |
| 返回值 | true：模型合法  false：模型不合法 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 遍历sum的所有模块 2. 检查sum模块的输入是否为空，如果为空，则返回false，并输出日志到控制台 3. 检查sum模块的输出是否为空，如果为空，则返回false，并输出日志到控制台。 4. 检查sum的输入模块是否存在，如果不存在，则返回false，并输出日志到控制台 5. 检查sum的输出模块是否存在，如果不存在，则返回false，并输出日志到控制台   6、若检查都通过则返回true |
| 人机交互界面设计 | 无 |

###### 检查Mult模块（ARMU1-SDD-PRASE-001-015）

checkMult函数见表 22。

表 22 checkMult函数

|  |  |
| --- | --- |
| 函数定义 | bool checkMult(); |
| 功能描述 | 检查Mult模块是否合法 |
| 输入 | m\_multModule |
| 输出 | 无 |
| 返回值 | true：模型合法  false：模型不合法 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 遍历mult的所有模块 2. 检查mult模块的输入是否为空，如果为空，则返回false，并输出日志到控制台 3. 检查mult模块的输出是否为空，如果为空，则返回false，并输出日志到控制台。 4. 检查mult的输入模块是否存在，如果不存在，则返回false，并输出日志到控制台 5. 检查mult的输出模块是否存在，如果不存在，则返回false，并输出日志到控制台   6、若检查都通过则返回true |
| 人机交互界面设计 | 无 |

###### 检查Disp模块（ARMU1-SDD-PRASE-001-016）

checkDisp函数见表 23。

表 23 checkDisp函数

|  |  |
| --- | --- |
| 函数定义 | bool checkDisp(); |
| 功能描述 | 检查Mult模块是否合法 |
| 输入 | m\_dispModule |
| 输出 | 无 |
| 返回值 | true：模型合法  false：模型不合法 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 遍历disp的所有模块 2. 检查disp模块的输入是否为空，如果为空，则返回false，并输出日志到控制台 3. 检查disp的输入模块是否存在，如果不存在，则返回false，并输出日志到控制台   4、若检查都通过则返回true |
| 人机交互界面设计 | 无 |

###### 检查Sine模块（ARMU1-SDD-PRASE-001-017）

checkSine函数见表 24。

表 24 checkSine函数

|  |  |
| --- | --- |
| 函数定义 | bool checkSine(); |
| 功能描述 | 检查Mult模块是否合法 |
| 输入 | m\_sineModule |
| 输出 | 无 |
| 返回值 | true：模型合法  false：模型不合法 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 遍历sine的所有模块 2. 检查sine模块的输出是否为空，如果为空，则返回false，并输出日志到控制台 3. 检查sine的输出模块是否存在，如果不存在，则返回false，并输出日志到控制台   4、若检查都通过则返回true |
| 人机交互界面设计 | 无 |

###### 检查步长是否合法（ARMU1-SDD-PRASE-001-018）

checkStep函数见表 25。

表 25 checkStep函数

|  |  |
| --- | --- |
| 函数定义 | bool checkStep(); |
| 功能描述 | 检查Mult模块是否合法 |
| 输入 | m\_step |
| 输出 | 无 |
| 返回值 | true：步长信息合法  false：步长信息不合法 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、检查步长是否大于0，若不大于0，返回false，并上报日志信息到控制台。  2、若检查都通过则返回true |
| 人机交互界面设计 | 无 |

###### 对模块的输入输出进行合法性检查（ARMU1-SDD-PRASE-001-019）

ioCheck函数见表 26。

表 26 ioCheck函数

|  |  |
| --- | --- |
| 函数定义 | bool ioCheck(const std::string\* io\_list, const int& list\_size); |
| 功能描述 | 检查模块的输入输出是否合法 |
| 输入 | io\_list：输入/输出列表  list\_size：输入/输出列表大小 |
| 输出 | 无 |
| 返回值 | true：输入/输出合法  false：输入/输出不合法 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、遍历io\_list  2、检查每一个输出/输入是否存在于m\_moduleList中，若不存在，则返回false  3、检查所有的输入/输出都合法后，返回true |
| 人机交互界面设计 | 无 |

###### 解析步长信息（ARMU1-SDD-PRASE-001-020）

parseStepSize函数见表 27。

表 27 parseStepSize函数

|  |  |
| --- | --- |
| 函数定义 | void parseStepSize(const rapidjson::Document& doc); |
| 功能描述 | 解析步长信息 |
| 输入 | const rapidjson::Document& doc：配置文件文本信息 |
| 输出 | m\_step |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、判断doc中是否有step字段的配置  2、提取仿真时长到m\_step中  3、提取仿真步长到m\_step中 |
| 人机交互界面设计 | 无 |

###### 记录模块的名称（ARMU1-SDD-PRASE-001-021）

recordModuleNumber函数见表 28。

表 28 recordModuleNumber函数

|  |  |
| --- | --- |
| 函数定义 | void recordModuleNumber(const std::string& name); |
| 功能描述 | 记录模块的名称 |
| 输入 | const std::string& name：模块名称 |
| 输出 | m\_moduleList |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、将输入name存入到m\_moduleList中 |
| 人机交互界面设计 | 无 |

###### 记录模块关系（ARMU1-SDD-PRASE-001-022）

recordRelationship函数见表 29。

表 29 recordRelationship函数

|  |  |
| --- | --- |
| 函数定义 | void recordRelationship(const std::string& from,const std::string& to); |
| 功能描述 | 记录模块之间的关系 |
| 输入 | const std::string& from：关系起始模块名称  const std::string& to：关系终点模块名称 |
| 输出 | m\_relationship |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 将关系起始模块名称from作为m\_relationship某一行数据的第一个元素 2. 将关系终点模块名称to作为m\_relationship某一行数据的第二个元素 |
| 人机交互界面设计 | 无 |

###### 获取模块ID（ARMU1-SDD-PRASE-001-023）

getModulePosition函数见表 30。

表 30 getModulePosition函数

|  |  |
| --- | --- |
| 函数定义 | int getModulePosition(std::string module\_name); |
| 功能描述 | 获取模块在m\_moduleList中的位置，即ID |
| 输入 | std::string module\_name：模块名称  m\_moduleList |
| 输出 | 无 |
| 返回值 | 模块ID |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 遍历m\_moduleList 2. 查找module\_name在m\_moduleList中的位置 3. 返回module\_name所在的位置 |
| 人机交互界面设计 | 无 |

###### 获取单例（ARMU1-SDD-PRASE-001-024）

getInstance函数见表 31。

表 31 getInstance函数

|  |  |
| --- | --- |
| 函数定义 | static ParseCfg& getInstance() |
| 功能描述 | 返回类的唯一实例 |
| 输入 | 无 |
| 输出 | 无 |
| 返回值 | ParseCfg类的唯一示例 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 返回ParseCfg类的唯一实例 |
| 人机交互界面设计 | 无 |

###### 解析模型配置文件入口（ARMU1-SDD-PRASE-001-025）

parseJsonConf函数见表 32。

表 32 parseJsonConf函数

|  |  |
| --- | --- |
| 函数定义 | bool parseJsonConf(); |
| 功能描述 | 解析模型配置文件的入口函数 |
| 输入 | 无 |
| 输出 | 无 |
| 返回值 | true：解析成功  false：解析失败 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 打开配置文件 2. 转换配置文件为文本类rapidjson::Document 3. 解析配置文件是否出错，若出错，则输出日志到控制台 4. 解析sin模块 5. 解析cons模块 6. 解析sun模块 7. 解析gain模块 8. 解析mult模块 9. 解析disp模块 10. 解析步长信息 11. 关闭配置文件 12. 对模型进行合法性检查，若检查出错则返回false   13、若以上都无问题，则返回true |
| 人机交互界面设计 | 无 |

#### 矩阵模块（ARMU1-SDD-MATRXI）

矩阵模块共包含1个.cpp文件，详细说明见表 33。

表 33 解析模块源文件列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 程序文件 | 概述 | 服务对象 | 函数个数 |
| 1 | matrix.cpp | 为模型关系创建零阶矩阵 | 仿真模块 | 8 |

##### matrix.cpp（ARMU1-SDD-MATRXI-001）

matrix.c的主要用途为模型关系创建零阶矩阵，该模块的简要描述见表 34，其函数列表见表 35。

表 34 matrix.cpp简要描述

|  |  |  |
| --- | --- | --- |
| 源文件名称 | 对外提供接口 | 涉及用例 |
| matrix.cpp | static Matrix& getInstance() | 仿真模型  支持成环计算 |
| void creatAdjacentMatrix(); |
| inline const std::vector<std::vector<int>>& getAdjacentMatrix() |

表 35 matrix.c函数列表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 函数定义 | 用途 | 对外接口 |
| 1 | static Matrix& getInstance() | 获取唯一单例 | 是 |
| 2 | void creatAdjacentMatrix() | 创建零阶矩阵 | 是 |
| 3 | inline const std::vector  <std::vector<int>>&getAdjacentMatrix() | 获取零阶矩阵 | 是 |
| 4 | Matrix() | 构造函数 | 否 |
| 5 | ~Matrix() | 析构函数 | 否 |
| 6 | int getModulePosition  (const std::string &module\_name) | 获取模块ID | 否 |

###### 构造函数（ARMU1-SDD-MATRXI-001-001）

Matrix函数见表 36

表 36 Matrix 函数

|  |  |
| --- | --- |
| 函数定义 | Matrix () |
| 功能描述 | 构造函数 |
| 输入 | m\_moduleList：模块列表 |
| 输出 | m\_adjacentMatrix：零阶矩阵 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1、遍历m\_moduleList，为m\_adjacentMatrix的矩阵全部赋值为0 |
| 人机交互界面设计 | 无 |

###### 析构函数（ARMU1-SDD-MATRXI-001-002）

~Matrix 函数见表 37

表 37 ~Matrix 函数

|  |  |
| --- | --- |
| 函数定义 | Matrix () |
| 功能描述 | 构造函数 |
| 输入 | 无 |
| 输出 | 无 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

###### 获取唯一单例（ARMU1-SDD-MATRXI-001-003）

getInstance函数见表 38

表 38 getInstance函数

|  |  |
| --- | --- |
| 函数定义 | static Matrix& getInstance() |
| 功能描述 | 获取矩阵模块类的唯一单例 |
| 输入 | 无 |
| 输出 | 无 |
| 返回值 | Matrix的单例 |
| 处理的全局变量 | 返回Matrix的唯一单例 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

###### 创建零阶矩阵（ARMU1-SDD-MATRXI-001-004）

creatAdjacentMatrix函数见表 39

表 39 creatAdjacentMatrix 函数

|  |  |
| --- | --- |
| 函数定义 | void creatAdjacentMatrix() |
| 功能描述 | 为模型关系创建零阶矩阵 |
| 输入 | m\_relationship：模型关系  m\_moduleList：模块列表 |
| 输出 | m\_adjacentMatrix：零阶矩阵 |
| 返回值 | 无 |
| 处理的全局变量 | 1. 遍历m\_relationship 2. 寻找m\_relationship每一组有关系的模块在m\_moduleList中的ID   3、将m\_adjacentMatrix位置为每一组有关系的模块的ID的值设置为1 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

###### 获取零阶矩阵（ARMU1-SDD-MATRXI-001-005）

getAdjacentMatrix函数见表 40

表 40 getAdjacentMatrix函数

|  |  |
| --- | --- |
| 函数定义 | inline const std::vector<std::vector<int>>&getAdjacentMatrix() |
| 功能描述 | 获取模型关系的零阶矩阵 |
| 输入 | 无 |
| 输出 | 无 |
| 返回值 | 零阶矩阵 |
| 处理的全局变量 | 返回零阶矩阵 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

###### 获取模块ID（ARMU1-SDD-MATRXI-001-006）

getModulePosition函数见表 41

表 41 getModulePosition函数

|  |  |
| --- | --- |
| 函数定义 | int getModulePosition(const std::string &module\_name) |
| 功能描述 | 获取模型在m\_moduleList中的ID |
| 输入 | const std::string &module\_name：模块名称  m\_moduleList：模块清单 |
| 输出 | 无 |
| 返回值 | 模块ID |
| 处理的全局变量 | 1. 遍历m\_moduleList 2. 寻找module\_name在m\_moduleList的位置，如果找到了则返回其ID   3、如果遍历完了，还未找到，则返回-1 |
| 处理主逻辑 | 无 |
| 人机交互界面设计 | 无 |

#### 仿真模块（ARMU1-SDD-SIMU）

仿真模块共包含1个.cpp文件，详细说明见表 33。

表 42 解析模块源文件列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 程序文件 | 概述 | 服务对象 | 函数个数 |
| 1 | simulation.cpp | 对模型进行仿真 | 无 | 8 |

##### simulation.cpp（ARMU1-SDD-SIMU-001）

simulation.cpp的主要用途为模型关系创建零阶矩阵，该模块的简要描述见表 43，其函数列表见表 44。

表 43 simulation.cpp简要描述

|  |  |  |
| --- | --- | --- |
| 源文件名称 | 对外提供接口 | 涉及用例 |
| simulation.c | static Simulation& getInstance() | 仿真模型  支持成环计算 |
| void startSimulation(); |
| void show(); |

表 44 simulation.cpp函数列表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 函数定义 | 用途 | 对外接口 |
| 1 | static Simulation& getInstance() | 获取仿真模块的唯一单例 | 是 |
| 2 | void startSimulation() | 开始模型仿真 | 是 |
| 3 | void show() | 展示数据信息 | 是 |
| 4 | double calculateSimulationResult  (const std::string& last\_model,  const double& sin\_value,  const bool& get\_pre\_value) | 计算仿真模型 | 否 |
| 5 | void startDfs(); | 开始深度优先搜索 | 否 |
| 6 | void dfs(int i); | 通过深度优先搜索寻找环 | 否 |
| 7 | Simulation | 构造函数 | 否 |
| 8 | ~Simulation | 析构函数 | 否 |

###### 构造函数（ARMU1-SDD-SIMU-001-001）

Simulation函数见表 45

表 45 Simulation函数

|  |  |
| --- | --- |
| 函数定义 | Simulation () |
| 功能描述 | 构造函数 |
| 输入 | data.csv文本 |
| 输出 | m\_color：颜色矩阵  m\_outFile：文件句柄 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 通过m\_outFile打开data.csv文本 2. 初始化m\_color都为0 |
| 人机交互界面设计 | 无 |

###### 析构函数（ARMU1-SDD-SIMU-001-002）

~Simulation函数见表 46

表 46 ~Simulation函数

|  |  |
| --- | --- |
| 函数定义 | ~Simulation () |
| 功能描述 | 构造函数 |
| 输入 | 无 |
| 输出 | m\_outFile：文件句柄 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 关闭m\_outFile句柄 |
| 人机交互界面设计 | 无 |

###### 获取仿真模块的唯一单例（ARMU1-SDD-SIMU-001-003）

getInstance函数见表 47

表 47 getInstance函数

|  |  |
| --- | --- |
| 函数定义 | static Simulation& getInstance() |
| 功能描述 | 获取仿真模块的唯一单例 |
| 输入 | 无 |
| 输出 | 无 |
| 返回值 | Simulation唯一实例 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 返回Simulation的唯一实例 |
| 人机交互界面设计 | 无 |

###### 开始模型仿真（ARMU1-SDD-SIMU-001-004）

startSimulation函数见表 48

表 48 startSimulation函数

|  |  |
| --- | --- |
| 函数定义 | void startSimulation() |
| 功能描述 | 开始仿真模型 |
| 输入 | m\_step  m\_dispModule |
| 输出 | 仿真结果输出到data.csv文本 |
| 返回值 | Simulation唯一实例 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 利用深度优先算法寻找代数换 2. 计算应该仿真的总步数 3. 开启循环，循环条件为仿真步数未达到第二步的总步数 4. 使用后序遍历计算模型结果 5. 仿真步数自增 |
| 人机交互界面设计 | 无 |

###### 展示数据信息（ARMU1-SDD-SIMU-001-005）

show函数见表 49

表 49 show函数

|  |  |
| --- | --- |
| 函数定义 | void show() |
| 功能描述 | 展示模型信息，如模型关系，零阶矩阵、模型参数、模型列表 |
| 输入 | 模型信息 |
| 输出 | 无 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 |  |
| 人机交互界面设计 | 无 |

###### 计算仿真模型（ARMU1-SDD-SIMU-001-006）

calculateSimulationResult函数见表 50

表 50 calculateSimulationResult函数

|  |  |
| --- | --- |
| 函数定义 | double calculateSimulationResult(const std::string& last\_model,  const double& sin\_value,  const bool& get\_pre\_value) |
| 功能描述 | 进行模型计算 |
| 输入 | const std::string& last\_model：当前模块名称  const double& sin\_value：sin模块的值  const bool& get\_pre\_value：是否为环  m\_cycle：代数环信息  m\_color：颜色矩阵  m\_gainModule：gain模块数据  m\_sumModule：sum模块数据  m\_multModule：mult模块数据  m\_sineModule：sine模块数据  m\_consModule：mult模块数据  m\_dispModule：disp模块数据 |
| 输出 | m\_gainModule：gain模块数据  m\_sumModule：sum模块数据  m\_multModule：mult模块数据  m\_sineModule：sine模块数据  m\_consModule：mult模块数据  m\_dispModule：disp模块数据 |
| 返回值 | 当前步当前模块的值 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 若当前模块为cons模块，则返回cons模块的参数值 2. 若当前模块为sum模块，则判断是否在环上并且为该环的起始模块，如果是起始模块则将上一步的sum的输出返回 3. 如果sum不环上并且为起始模块，则遍历sum的两个输入 4. 遍历m\_cycle，查看sum是否和其它模块构成环，如果是，则退出当前遍历 5. 若遍历结束后，只遍历到了第一个输入，则说明这个输入和sum构成了环，并且是该输入指向sum的，则进行两个输入的递归相加，并将第一个输入的get\_pre\_value设置为true 6. 若遍历结束后，只遍历到了第二个输入，则说明这个输入和sum构成了环，并且是该输入指向sum的，则进行两个输入的递归相加，并将第一个输入的get\_pre\_value设置为true 7. 若遍历结束后，未检测到环，则进行两个输入的递归相加。 8. 返回m\_sumModule的输出 9. 若当前的模块为gain模块，则判断是否在环上并且为该环的起始模块，如果是起始模块则将上一步的gain的输出返回 10. 若gain不在环上，则计算gain模块当前步的输出值，并返回其输出值 11. 若当前模块为sine模块，则计算sine模块当前步的值 12. 若当前模块为mult模块，则判断是否在环上并且为该环的起始模块，如果是起始模块则将上一步的mult的输出返回 13. 如果mult不环上并且为起始模块，则遍历mult的两个输入 14. 遍历m\_cycle，查看mult是否和其它模块构成环，如果是，则退出当前遍历 15. 若遍历结束后，只遍历到了第一个输入，则说明这个输入和mult构成了环，并且是该输入指向sum的，则进行两个输入的递归相加，并将第一个输入的get\_pre\_value设置为true 16. 若遍历结束后，只遍历到了第二个输入，则说明这个输入和mult构成了环，并且是该输入指向mult的，则进行两个输入的递归相加，并将第一个输入的get\_pre\_value设置为true 17. 若遍历结束后，未检测到环，则进行两个输入的递归相加。 18. 返回m\_multModule的输出 |
| 人机交互界面设计 | 无 |

###### 开始深度优先搜索（ARMU1-SDD-SIMU-001-007）

startDfs函数见表 51

表 51 startDfs函数

|  |  |
| --- | --- |
| 函数定义 | void startDfs(); |
| 功能描述 | 对模型进行深度优先搜索 |
| 输入 | m\_moduleList：模块列表 |
| 输出 | m\_cycle：代数环信息  m\_color：颜色矩阵 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 遍历模块列表 2. 若当前模块对应在颜色矩阵的值为0，则进行深度优先搜索找代数环 |
| 人机交互界面设计 | 无 |

###### 通过深度优先搜索寻找环（ARMU1-SDD-SIMU-001-008）

dfs函数见表 52

表 52 dfs函数

|  |  |
| --- | --- |
| 函数定义 | void dfs(int i); |
| 功能描述 | 对模型进行深度优先搜索 |
| 输入 | int i：模块ID  m\_moduleList：模块列表 |
| 输出 | m\_cycle：代数环信息  m\_color：颜色矩阵 |
| 返回值 | 无 |
| 处理的全局变量 | 无 |
| 处理主逻辑 | 1. 设置当前模块的颜色为-1，即表示该模块已经遍历过了 2. 获取零阶矩阵 3. 遍历模块列表 4. 若当前模块和其他模块在零阶矩阵中的值不为0，则判断遍历模块列表的模块对应颜色矩阵的值是否-1，若为-1则说明该模块已经和别人构成环了，将环信息存入到m\_cycle中 5. 若遍历模块列表对应颜色矩阵的值为0，说明未被遍历过，则递归往下遍历 6. 全部遍历结束后，设置当前模块（即i对应的模块）对应颜色矩阵中的值为1（即该模块包括该模块后续的所有模块都遍历过了） |
| 人机交互界面设计 | 无 |

* + 1. 执行方案

该软件采用模块化设计，模块分为模型数据、解析模块、矩阵模块、仿真模块。四个模块之间相辅相成，共同完成仿真计算任务。模块间的数据流如图 3所示



图 3 模块间数据流

仿真软件所使用的模型为json格式的模型配置文件，里面描述了模型的信息以及模型之间的关系。解析模块将配置文件解析后，将解析后的数据保存到模型数据中，供矩阵模块和仿真模块使用模型数据。在保存模型数据后，矩阵模块根据模块列表和模块关系给模块创建模块关系的零阶矩阵，在仿真模块进行仿真时，将根据零阶矩阵先寻找环的位置，然后根据后续遍历对模型进行递归计算，计算出每一个模块在当前步的输出，对于成环的模块，将使用其上一个步长的值，其序列图如图 4所示。



图 4 执行方案序列图

* + 1. 接口设计
       1. 内部接口
          1. 接口标识和接口图

仿真软件内部采用模块化设计，共涉及四个模块，分别为解析模块、模型数据、矩阵模块和仿真模块，其中解析模块、矩阵模块和仿真模块采用单例设计，保证全局的数据都是仅此一份。模型数据作为公共数据库为其他模块提供数据支持。各模块之间的接口如图 5所示，各模块之间的接口列表如表 53所示。



图 5 接口关系

表 53软件外部接口描述表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名称 | 接口标识 | 接口类型 | 接口用途 |
| 1/2/3 | 步长信息 | ARMU1-SDD-I/F-001 | 共享数据 | 保存模型步长信息 |
| Gain模块 | ARMU1-SDD-I/F-002 | 共享数据 | 保存Gain模块信息 |
| Sum模块 | ARMU1-SDD-I/F-003 | 共享数据 | 保存Sum模块信息 |
| Mult模块 | ARMU1-SDD-I/F-004 | 共享数据 | 保存Mult模块信息 |
| Sine模块 | ARMU1-SDD-I/F-005 | 共享数据 | 保存Sine模块信息 |
| Cons模块 | ARMU1-SDD-I/F-006 | 共享数据 | 保存Cons模块信息 |
| Disp模块 | ARMU1-SDD-I/F-007 | 共享数据 | 保存Disp模块信息 |
| 模块列表 | ARMU1-SDD-I/F-008 | 共享数据 | 保存所有模块的名称 |
| 模块关系 | ARMU1-SDD-I/F-009 | 共享数据 | 保存所有模块的关系 |
| 4 | 获取零阶矩阵 | ARMU1-SDD-I/F-010 | 类接口 | 获取零阶矩阵信息 |

* + - * 1. 步长信息 （ARMU1-SDD-I/F-001）

步长信息接口需求见表 54。

表 54步长信息接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 步长信息 | | 接口标识 | ARMU1-SDD-I/F-001 |
| 内部接口实体 | 模型数据 | | 接口类型 | 共享数据 |
| 接口用途 | 保存模型的步长信息 | | | |
| 接口数据 | 1. double final\_time：仿真时长 2. double step\_size：仿真步长 | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - * 1. Gain模块 （ARMU1-SDD-I/F-002）

Gain模块接口需求见表 55。

表 55Gain模块接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | Gain模块 | | 接口标识 | ARMU1-SDD-I/F-002 |
| 内部接口实体 | 模型数据 | | 接口类型 | 共享数据 |
| 接口用途 | 保存Gain模块信息 | | | |
| 接口数据 | 接口定义：static std::map<std::string, Gain> m\_gainModule  成员数据：  ModuleInfo info {  std::string input[2] ：输入模块名称  std::map<std::string, double> output：输出模块名称及值  double output\_value：gain模块输出值  } // 模型IO信息  double gain\_value：gain增益大小 | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - * 1. Sum模块 （ARMU1-SDD-I/F-003）

Sum模块接口需求见表 56。

表 56Sum模块接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | Sum模块 | | 接口标识 | ARMU1-SDD-I/F-003 |
| 内部接口实体 | 模型数据 | | 接口类型 | 共享数据 |
| 接口用途 | 保存Sum模块信息 | | | |
| 接口数据 | 接口定义：static std::map<std::string, ModuleInfo> m\_sumModule  成员数据：  ModuleInfo info {  std::string input[2] ：输入模块名称  std::map<std::string, double> output：输出模块名称及值  double output\_value：模块输出值  } // 模型IO信息 | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - * 1. Mult模块（ ARMU1-SDD-I/F-004）

Mult模块接口需求见表 57。

表 57 Mult模块接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | Mult模块 | | 接口标识 | ARMU1-SDD-I/F-004 |
| 内部接口实体 | 模型数据 | | 接口类型 | 共享数据 |
| 接口用途 | 保存Mult模块信息 | | | |
| 接口数据 | 接口定义：static std::map<std::string, ModuleInfo> m\_multModule  成员数据：  ModuleInfo info {  std::string input[2] ：输入模块名称  std::map<std::string, double> output：输出模块名称及值  double output\_value：模块输出值  } // 模型IO信息 | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - * 1. Sine模块（ ARMU1-SDD-I/F-005）

Sine模块接口需求见表 58。

表 58 Sine模块接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | Sine模块 | | 接口标识 | ARMU1-SDD-I/F-005 |
| 内部接口实体 | 模型数据 | | 接口类型 | 共享数据 |
| 接口用途 | 保存Sine模块信息 | | | |
| 接口数据 | 接口定义：static std::map<std::string, Sine> m\_sineModule  成员数据：  typedef struct Sine {  std::map<std::string, double> output：模块输出  double amplitude：幅值  double output\_value：输出值  } Sine; | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - * 1. Cons模块（ ARMU1-SDD-I/F-006）

Cons模块接口需求见表 59。

表 59 Cons模块接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | Cons模块 | | 接口标识 | ARMU1-SDD-I/F-006 |
| 内部接口实体 | 模型数据 | | 接口类型 | 共享数据 |
| 接口用途 | 保存Cons模块信息 | | | |
| 接口数据 | 接口定义：static std::map<std::string, Cons> m\_consModule  成员数据：  typedef struct Cons {  std::map<std::string, double> output：模型输出列表  double cons\_value：常量值  double output\_value：cons输出的值  } Cons; | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - * 1. Disp模块 （ARMU1-SDD-I/F-007）

Disp模块接口需求见表 60。

表 60 Disp模块接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | Disp模块 | | 接口标识 | ARMU1-SDD-I/F-007 |
| 内部接口实体 | 模型数据 | | 接口类型 | 共享数据 |
| 接口用途 | 保存Disp模块信息 | | | |
| 接口数据 | 接口定义：static std::map<std::string, Disp> m\_dispModule  成员数据：  typedef struct Disp {  std::map<std::string, double> input：模型输入  } Disp; | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - * 1. 模块列表 （ARMU1-SDD-I/F-008）

模块列表接口需求见表 61。

表 61 模块列表接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 模块列表 | | 接口标识 | ARMU1-SDD-I/F-008 |
| 内部接口实体 | 模型数据 | | 接口类型 | 共享数据 |
| 接口用途 | 保存模块列表信息 | | | |
| 接口数据 | 接口定义：static std::vector<std::string> m\_moduleList | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - * 1. 模块关系 （ARMU1-SDD-I/F-009）

模块关系接口需求见表 62。

表 62 模块关系接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 模块关系 | | 接口标识 | ARMU1-SDD-I/F-009 |
| 内部接口实体 | 模型数据 | | 接口类型 | 共享数据 |
| 接口用途 | 保存模块关系 | | | |
| 接口数据 | 接口定义：static std::vector<std::vector<std::string>> m\_relationship | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - * 1. 获取零阶矩阵 （ARMU1-SDD-I/F-010）

获取零阶矩阵接口需求见表 63。

表 63获取零阶矩阵接口需求表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 接口名称 | 获取零阶矩阵 | | 接口标识 | ARMU1-SDD-I/F-010 |
| 内部接口实体 | 矩阵模块 | | 接口类型 | 共享数据 |
| 接口用途 | 获取零阶矩阵信息 | | | |
| 接口数据 | std::vector<std::vector<int>> m\_adjacentMatrix：零阶矩阵 | | | |
| 接口通信特征 | 通信链路 | 共享数据 | | |
| 数据传输 | 非周期，按需获取 | | |
| 接口协议特征 | 无 | | | |

* + - 1. 外部接口

仿真软件对外使用的接口为json文件，json文件描述了模型所含有的模块名称、模块参数、模块的输入输出关系，只要名称不一样，同类型的模块支持多个模块一起使用。其json格式如下所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 模块种类 | 模块参数 | 参数名称 | 参数类型 | 含义 |
| 1 | gain | name | 模块名称 | 字符串 | 表示模块的名称 |
| value | 增益值 | 双精度浮点数 | 表示放大的倍数 |
| input | 输入 | 字符串数组 | 表示输入的模块名称 |
| output | 输出 | 字符串数组 | 表示输出的模块名称 |
| intivalue | 初始值 | 双精度浮点数 | 表示其初始值 |
| 2 | cons | name | 模块名称 | 字符串 | 表示模块的名称 |
| value | 常量值 | 双精度浮点数 | 表示常量值 |
| output | 输出 | 字符串数组 | 表示输出的模块名称 |
| 3 | mult | name | 模块名称 | 字符串 | 表示模块的名称 |
| input | 输入 | 字符串数组 | 表示输入的模块名称 |
| output | 输出 | 字符串数组 | 表示输出的模块名称 |
| intivalue | 初始值 | 双精度浮点数 | 表示其初始值 |
| 4 | sum | name | 模块名称 | 字符串 | 表示模块的名称 |
| input | 输入 | 字符串数组 | 表示输入的模块名称 |
| output | 输出 | 字符串数组 | 表示输出的模块名称 |
| intivalue | 初始值 | 双精度浮点数 | 表示其初始值 |
| 5 | disp | name | 模块名称 | 字符串 | 表示模块的名称 |
| input | 输入 | 字符串数组 | 表示输入的模块名称 |
| 6 | sine | name | 模块名称 | 字符串 | 表示模块的名称 |
| value | 幅值 | 双精度浮点数 | 表示sin的幅值 |
| output | 输出 | 字符串数组 | 表示输出的模块名称 |
| 7 | step | finaltime | 仿真时长 | 双精度浮点数 | 表示仿真的时长 |
|  |  | stepsize | 仿真步长 | 双精度浮点数 | 表示仿真的步长 |

示例json如下：

{

"gain": [

{

"name": "gain1",

"value": 2.0,

"input": [ "cons1" ],

"output": [ "sum1" ]

}

],

"sum": [

{

"name": "sum1",

"input": [ "sine1", "gain1" ],

"output": [ "disp1" ]

}

],

"cons": [

{

"name": "cons1",

"value": 2.0,

"output": [ "gain1" ]

}

],

"disp": [

{

"name": "disp1",

"input": [ "sum1" ]

}

],

"sine": [

{

"name": "sine1",

"value": 2.0,

"output": [ "sum1" ]

}

],

"step": [

{

"finaltime": 10.0,

"stepsize": 0.01

}

]

}

* 1. CSCI详细设计
     1. 仿真模型

1）执行状态

程序启动后

2）触发条件

仿真软件运行后

3）功能描述

进行模型的仿真计算。模型的仿真计算通过后序遍历的方式进行递归，将当前步的每一个模块的值计算出来，其函数内部的实现流程如图 6所示。



图 6 仿真模型流程

4）输入

详见表 50

5）输出

详见表 50

* 1. 需求可追踪性

略。

* 1. 注解

本章应包括有助于了解本文档的所有信息（例如，背景、术语、缩略语或公式）。