(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 118607222 A
(43)申请公布日 2024.09.06

(21)申请号 202410737676.2

(22)申请日 2024.06.07

(71)申请人 深圳景元数宇科技有限公司
地址 518000 广东省深圳市福田区福保街道福保社区市花路南侧长富金茂大厦1号楼3103

(72)发明人 何晓龙 刘宇超 杨浩 刘志伟 周凡利

(74)专利代理机构 成都顶峰专利事务所(普通合伙) 51224
专利代理师 王袁辉

(51)Int.Cl.
G06F 30/20(2020.01)
G06F 8/30(2018.01)
G06F 9/448(2018.01)

权利要求书2页 说明书8页 附图2页

(54)发明名称

基于Python模型库的ModeLica建模方法、装置、电子设备及产品

(57)摘要

本发明公开了一种基于Python模型库的Modelica建模方法、装置、电子设备及产品,本发明利用Sysplorer建模工具可支持Modelica与Python文件同时建模这一特性,来以Sysplorer建模工具为基础,实现Modelica与Python的联合建模;其中,在建模时,先将PythonInterface模型库导入至Sysplorer建模工具内,然后再生成Sysplorer模型;而后,利用导入的PythonInterface模型库,生成Sysplorer模型的Python实例化组件,从而基于该Python实例化组件来实现Python文件的调用;接着,根据调用的Python文件,生成Sysplorer模型的Modelica文本信息;最后,则可基于此,来构建出Modelica模型;如此,本发明在整个建模过程中,可直接在Modelica环境中使用python文件来完成建模,且整个过程由人工操作变为自动化运行;由此,不仅减少了模型开发人员的工作量,且还提升了建模效率。

CN 118607222 A

1.一种基于Python模型库的Modelica建模方法,其特征在于,包括:

获取PythonInterface模型库,并将所述PythonInterface模型库导入至建模工具内,其中,所述建模工具包括Sysplorer建模工具;

基于所述建模工具,生成Sysplorer模型,并利用所述建模工具内的PythonInterface模型库,生成所述Sysplorer模型的Python实例化组件;

获取所述Python实例化组件的python文件,并基于所述python文件,生成所述Sysplorer模型的Modelica文本信息;

利用所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型。

2.根据权利要求1所述的方法,其特征在于,利用所述建模工具内的PythonInterface模型库,生成所述Sysplorer模型的Python实例化组件,包括:

将所述建模工具内的PythonInterface模型库中的Python对象调用组件配置进所述Sysplorer模型中,以得到配置后的Sysplorer模型;

对所述配置后的Sysplorer模型进行实例化组件处理,以在实例化组件处理后,得到所述Python实例化组件。

3.根据权利要求1所述的方法,其特征在于,基于所述python文件,生成所述Sysplorer模型的Modelica文本信息,包括:

对所述python文件进行解析处理,以得到所述python文件中的输入参数、输出参数以及参数总数量,其中,所述输入参数和所述输出参数为建模参数;

确定出所述python文件中每个输入参数和每个输出参数的数据类型;

基于所述python文件,以及所述python文件中每个输入参数的数据类型、每个输出参数的数据类型以及参数总数量,生成所述Sysplorer模型的Modelica文本信息。

4.根据权利要求3所述的方法,其特征在于,对所述python文件进行解析处理,以得到所述python文件中的输入参数、输出参数以及参数总数量,包括:

对所述python文件进行解析处理,得到所述python文件中的所有python类;

判断各python类中是否包含有预设函数,其中,所述预设函数包括setp_impl函数;

若是,则获取各个预设函数的参数列表;

基于各个参数列表,确定出所述python文件中的输入参数、输出参数以及参数总数量。

5.根据权利要求4所述的方法,其特征在于,若任一预设函数中未含有所述预设函数,则所述方法还包括:

生成错误提示信息,并以弹窗形式将所述错误提示信息进行可视化展示。

6.根据权利要求3所述的方法,其特征在于,基于所述python文件,以及所述python文件中每个输入参数的数据类型、每个输出参数的数据类型以及参数总数量,生成所述Sysplorer模型的Modelica文本信息,包括:

获取所述python文件的文件路径和环境依赖路径;

基于所述文件路径、所述环境依赖路径、所述参数总数量、每个输入参数的数据类型以及每个输出参数的数据类型,生成所述Sysplorer模型的Modelica文本信息。

7.根据权利要求1所述的方法,其特征在于,利用所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型,包括:

获取Modelica仿真函数库;

基于所述Modelica仿真函数库和所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型。

8.一种基于Python模型库的Modelica建模装置,其特征在于,包括:

库导入单元,用于获取PythonInterface模型库,并将所述PythonInterface模型库导入至建模工具内,其中,所述建模工具包括Sysplorer建模工具;

实例化单元,用于基于所述建模工具,生成Sysplorer模型,并利用所述建模工具内的PythonInterface模型库,生成所述Sysplorer模型的Python实例化组件;

Modelica文本信息生成单元,用于获取所述Python实例化组件的python文件,并基于所述python文件,生成所述Sysplorer模型的Modelica文本信息;

模型生成单元,用于利用所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型。

9.一种电子设备,其特征在于,包括:依次通信相连的存储器、处理器和收发器,其中,所述存储器用于存储计算机程序,所述收发器用于收发消息,所述处理器用于读取计算机程序,执行如权利要求1~7任意一项所述的基于Python模型库的Modelica建模方法。

10.一种包含指令的计算机程序产品,其特征在于,当指令在计算机上运行时,使计算机执行如权利要求1~7任意一项所述的基于Python模型库的Modelica建模方法。

基于Python模型库的ModeLica建模方法、装置、电子设备及产品

技术领域

[0001] 本发明属于软件工程处理技术领域,具体涉及一种基于Python模型库的Modelica建模方法、装置、电子设备及产品。

背景技术

[0002] 由于目前有很多算法都是基于Python开发的,因此,各个建模环境大多都会结合Python语言进行使用;其中,对于Modelica建模也是如此;但是,在实际应用过程中,要在Modelica环境(是一种开放、面向对象、基于方程的计算机语言环境)中使用Python语言,则需要用户知晓Python的底层算法逻辑和调用关系,然后手动编译相应的代码,并将其转换为Modelica语言,才能实现其在Modelica环境中的应用;如此,不仅会增加工作人员的工作量,且还降低了建模效率;基于此,如何提供一种可在Modelica环境中直接使用Python语言,来提高建模效率以及减少建模工作量的建模方法,已成为了一个亟待解决的问题。

发明内容

[0003] 本发明的目的是提供一种基于Python模型库的Modelica建模方法、装置、电子设备及产品,用以解决现有技术所存在的建模工作量大以及效率低的问题。

[0004] 为了实现上述目的,本发明采用以下技术方案:

[0005] 第一方面,提供了一种基于Python模型库的Modelica建模方法,包括:

[0006] 获取PythonInterface模型库,并将所述PythonInterface模型库导入至建模工具内,其中,所述建模工具包括Sysplorer建模工具;

[0007] 基于所述建模工具,生成Sysplorer模型,并利用所述建模工具内的PythonInterface模型库,生成所述Sysplorer模型的Python实例化组件;

[0008] 获取所述Python实例化组件的python文件,并基于所述python文件,生成所述Sysplorer模型的Modelica文本信息;

[0009] 利用所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型。

[0010] 基于上述公开的内容,本发明利用Sysplorer建模工具可支持Modelica与Python文件同时建模这一特性,来以Sysplorer建模工具为基础,实现Modelica与Python的联合建模;其中,在建模时,先将PythonInterface模型库导入至Sysplorer建模工具内,然后再基于Sysplorer建模工具生成Sysplorer模型;而后,再利用导入的PythonInterface模型库,生Sysplorer模型的Python实例化组件,从而基于该Python实例化组件来实现Python文件的调用;接着,本发明在调用的Python文件的基础上,生成Sysplorer模型的Modelica文本信息;最后,基于Modelica文本信息,并利用Sysplorer建模工具,即可生成Sysplorer模型对应的Modelica模型。

[0011] 通过上述设计,本发明通过建立Python-Modelica联合建模的方式,使得可在Modelica环境中直接使用python算法来完成建模,且整个过程由人工操作变为自动化运

行,如此,不仅降低了出错概率,减少了模型开发人员的工作量,且还提升了建模效率;因此,非常适用于大规模应用与推广。

[0012] 在一个可能的设计中,利用所述建模工具内的PythonInterface模型库,生成所述Sysplorer模型的Python实例化组件,包括:

[0013] 将所述建模工具内的PythonInterface模型库中的Python对象调用组件配置进所述Sysplorer模型中,以得到配置后的Sysplorer模型;

[0014] 对所述配置后的Sysplorer模型进行实例化组件处理,以在实例化组件处理后,得到所述Python实例化组件。

[0015] 在一个可能的设计中,基于所述python文件,生成所述Sysplorer模型的Modelica文本信息,包括:

[0016] 对所述python文件进行解析处理,以得到所述python文件中的输入参数、输出参数以及参数总数量,其中,所述输入参数和所述输出参数为建模参数;

[0017] 确定出所述python文件中每个输入参数和每个输出参数的数据类型;

[0018] 基于所述python文件,以及所述python文件中每个输入参数的数据类型、每个输出参数的数据类型以及参数总数量,生成所述Sysplorer模型的Modelica文本信息。

[0019] 在一个可能的设计中,对所述python文件进行解析处理,以得到所述python文件中的输入参数、输出参数以及参数总数量,包括:

[0020] 对所述python文件进行解析处理,得到所述python文件中的所有python类;

[0021] 判断各python类中是否包含有预设函数,其中,所述预设函数包括setp_impl函数;

[0022] 若是,则获取各个预设函数的参数列表;

[0023] 基于各个参数列表,确定出所述python文件中的输入参数、输出参数以及参数总数量。

[0024] 在一个可能的设计中,若任一预设函数中未含有所述预设函数,则所述方法还包括:

[0025] 生成错误提示信息,并以弹窗形式将所述错误提示信息进行可视化展示。

[0026] 在一个可能的设计中,基于所述python文件,以及所述python文件中每个输入参数的数据类型、每个输出参数的数据类型以及参数总数量,生成所述Sysplorer模型的Modelica文本信息,包括:

[0027] 获取所述python文件的文件路径和环境依赖路径;

[0028] 基于所述文件路径、所述环境依赖路径、所述参数总数量、每个输入参数的数据类型以及每个输出参数的数据类型,生成所述Sysplorer模型的Modelica文本信息。

[0029] 在一个可能的设计中,利用所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型,包括:

[0030] 获取Modelica仿真函数库;

[0031] 基于所述Modelica仿真函数库和所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型。

[0032] 第二方面,提供了一种基于Python模型库的Modelica建模装置,包括:

[0033] 库导入单元,用于获取PythonInterface模型库,并将所述PythonInterface模型

库导入至建模工具内,其中,所述建模工具包括Sysplorer建模工具;

[0034] 实例化单元,用于基于所述建模工具,生成Sysplorer模型,并利用所述建模工具内的PythonInterface模型库,生成所述Sysplorer模型的Python实例化组件;

[0035] Modelica文本信息生成单元,用于获取所述Python实例化组件的python文件,并基于所述python文件,生成所述Sysplorer模型的Modelica文本信息;

[0036] 模型生成单元,用于利用所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型。

[0037] 第三方面,提供了另一种基于Python模型库的Modelica建模装置,以装置为电子设备为例,包括依次通信相连的存储器、处理器和收发器,其中,所述存储器用于存储计算机程序,所述收发器用于收发消息,所述处理器用于读取所述计算机程序,执行如第一方面或第一方面中任意一种可能设计的所述基于Python模型库的Modelica建模方法。

[0038] 第四方面,提供了一种存储介质,存储介质上存储有指令,当所述指令在计算机上运行时,执行如第一方面或第一方面中任意一种可能设计的所述基于Python模型库的Modelica建模方法。

[0039] 第五方面,提供了一种包含指令的计算机程序产品,当所述指令在计算机上运行时,使计算机执行如第一方面或第一方面中任意一种可能设计的所述基于Python模型库的Modelica建模方法。

[0040] 有益效果:

[0041] (1)本发明通过建立Python-Modelica联合建模的方式,使得可在Modelica环境中直接使用python算法来完成建模,且整个过程由人工操作变为自动化运行,如此,不仅降低了出错概率,减少了模型开发人员的工作量,且还提升了建模效率;因此,非常适用于大规模应用与推广。

附图说明

[0042] 图1为本发明实施例提供的基于Python模型库的Modelica建模方法的步骤流程示意图;图2为本发明实施例提供的基于Python模型库的Modelica建模装置的结构示意图;

[0043] 图3为本发明实施例提供的电子设备的结构示意图。

具体实施方式

[0044] 为了更清楚地说明本发明实施例或现有技术中的技术方案,下面将结合附图和实施例或现有技术的描述对本发明作简单地介绍,显而易见地,下面关于附图结构的描述仅仅是本发明的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。在此需要说明的是,对于这些实施例方式的说明用于帮助理解本发明,但并不构成对本发明的限定。

[0045] 应当理解,尽管本文可能使用术语第一、第二等等来描述各种单元,但是这些单元不应当受到这些术语的限制。这些术语仅用于区分一个单元和另一个单元。例如可以将第一单元称作第二单元,并且类似地可以将第二单元称作第一单元,同时不脱离本发明的示例实施例的范围。

[0046] 应当理解,对于本文中可能出现的术语“和/或”,其仅仅是一种描述关联对象的关

联关系,表示可以存在三种关系,例如,A和/或B,可以表示:单独存在A,单独存在B,同时存在A和B三种情况;对于本文中可能出现的术语“/和”,其是描述另一种关联对象关系,表示可以存在两种关系,例如,A/和B,可以表示:单独存在A,单独存在A和B两种情况;另外,对于本文中可能出现的字符“/”,一般表示前后关联对象是一种“或”关系。

[0047] 实施例:

[0048] 参见图1所示,本实施例所提供的基于Python模型库的Modelica建模方法,利用Sysplorer建模工具可支持Modelica与Python文件同时建模这一特性,来以Sysplorer建模工具为基础,实现Modelica与Python的联合建模;其中,在建模时,先将PythonInterface模型库导入至Sysplorer建模工具内,然后再生成Sysplorer模型;而后,利用导入的PythonInterface模型库,生成Sysplorer模型的Python实例化组件,从而基于该Python实例化组件来实现Python文件的调用;接着,根据调用的Python文件,生成Sysplorer模型的Modelica文本信息;最后,则可基于此,来构建出Modelica模型;如此通过上述设计,本方法在整个建模过程中,可直接在Modelica环境中使用python文件来完成建模,且整个过程由人工操作变为自动化运行;由此,不仅减少了模型开发人员的工作量,且还提升了建模效率;因此,非常适用于大规模应用与推广;其中,举例本方法可以但不限于在建模端侧运行,可选的,建模端可以但不限于为个人电脑,可以理解的,前述执行主体并不构成对本申请实施例的限定,相应的,本方法的运行步骤可以但不限于如下述步骤S1～S4所示。

[0049] S1.获取PythonInterface模型库,并将所述PythonInterface模型库导入至建模工具内,其中,所述建模工具包括Sysplorer建模工具;在本实施例中,可预先在建模端内设置PythonInterface模型库,从而在使用时,直接调用该模型库,并将其导入至Sysplorer建模工具内即可;进一步的,本实施例是利用Sysplorer建模工具支持Modelica与Python文件同时建模的这一特性,来实现在Modelica环境中,直接使用Python文件进行建模的功能;其中,Sysplorer建模工具用于提供界面编辑功能,其集成在MWORKS.Sysplorer环境中,且提供了2个右键菜单选项和2个python环境配置界面,如此,基于该Sysplorer建模工具,则可使用Python文件,来自动生成Modelica文本并完成建模。

[0050] 其中,两个右键菜单选项用于提供用户可选操作,通过选择不同的项进入不同的对话框,以编辑模型,具体的,可以包括:(1)选择Python对象,弹出python文件选择对话框,以供用户选择python文件以及其依赖文件路径;(2)编辑PythonObject端口:其可弹出“编辑PythonObject端口”的对话框,以根据用户所选择的python文件,选择用户所需的python类,并根据该类中的step_impl函数,配置输入输出端口信息;同理,两个环境配置界面则用于导入Python对象和编辑PythonObject端口信息,其中,前者用于导入python文件,配置python运行环境和选择python依赖文件,后者则用于导入需要使用的python类,配置的输入输出端口信息,并会将此次的配置信息写入到Modelica文本中,可选的,主要是记录在文本内部的inputDims、inputTypes、outputDims、outputTypes字段中。

[0051] 同时,PythonInterface模型库是基于Modelica语言为支持python模块做的标准模型,其规定了Modelica与Python同时建模的标准规范,交互过程中的初始化接口,单步计算接口,结束调用接口等一系列标准,并支持了Python环境中的Real、Integer、Boolean、Npy_Int、Npy_Int、Npy_Bool、Python Object等数据类型的传输。

[0052] 更进一步的,PythonInterface模型库包括:constructor(构造器)模块、

exchangeData模块(一种基于Python编写的库,主要是帮助开发者进行数据获取、转换、存储)、destructor(析构函数)模块、CalcRealPortNumber模块和CalcIntPortNumber模块;其中,constructor模块位于模型库中的PythonInterface.Communication.PythonObject.constructor内,其功能是:以外部函数的形式,调用了“my_python_object.dll”动态链接库,并使用了initPythonMemory函数。

[0053] exchangeData模块位于PythonInterface.Communication.PythonObject.exchangeData内,其功能是:以外部函数的形式,调用了“my_python_object.dll”动态链接库,并使用stepImpl函数。

[0054] destructor模块位于PythonInterface.Communication.PythonObject.destructor内,其功能是:以外部函数的形式,调用了“my_python_object.dll”动态链接库,并使用freePythonMemory函数。

[0055] 接着,CalcRealPortNumber模块则位于:

[0056] PythonInterface.Communication.PythonObject.CalcRealPortNumber内,其功能是通过modelica中outputDims、outputTypes、hasOutput字段内容,计算出所需要的Integer类型的端口数量。

[0057] 同理,CalcIntPortNumber模块则位于:

[0058] PythonInterface.Communication.PythonObject.CalcIntPortNumber内,其功能是:通过modelica中outputDims、outputTypes、hasOutput字段内容,计算出所需要的Integer类型的端口数量。

[0059] 如此,基于该PythonInterface模型库以及Sysplorer建模工具,则可实现Modelica与Python之间的直接转换。

[0060] 在具体实施时,完成PythonInterface模型库的导入后,则可先生成Sysplorer模型,并建立该模型的Python实例化组件,以便后续基于该实例化组件,来调用Python文件;其中,Sysplorer模型的生成过程及实例化组件的生成过程可以但不限于如下述步骤S2所示。

[0061] S2.基于所述建模工具,生成Sysplorer模型,并利用所述建模工具内的PythonInterface模型库,生成所述Sysplorer模型的Python实例化组件;在具体实施时,Sysplorer模型可以但不限预先设置在建模端内,使用时调用即可;当然,也可在响应于建模人机交互操作时,基于该人机交互操作生成;同时,在建立得到Sysplorer模型后,则可生成其对应的Python实例化组件;其中,本实施例是先将所述建模工具内的PythonInterface模型库中的Python对象调用组件配置进所述Sysplorer模型中,从而得到配置后的Sysplorer模型;然后,再对对所述配置后的Sysplorer模型进行实例化组件处理,以在实例化组件处理后,得到所述Python实例化组件。

[0062] 在具体实施时,举例前述Python对象调用组件可以但不限于采用:PythonInterface.PythonObject.PythonObject模块,即将PythonInterface模型库中的PythonInterface.PythonObject.PythonObject模块拖入用户模型(也就是前述Sysplorer模型),并进行实例化组件处理,从而得到Python实例化组件(当然也得到组件名称,如base1);在本实施例中,实例化组件的目的是为了实现Python文件的调用,其是Python语言中的常用技术,其原理不再赘述。

[0063] 在得到Sysplorer模型的Python实例化组件后,则可基于此,来进行对应Python文件的调用,以便基于调用的Python文件,来生成Modelica文本信息,其中,文件调用以及Modelica文本信息生成过程可以但不限于如下述步骤S3所示。

[0064] S3.获取所述Python实例化组件的python文件,并基于所述python文件,生成所述Sysplorer模型的Modelica文本信息;在具体实施时,举例可以但不限于响应于Python实例化组件的Python对象选择人机交互操作,以从调用的python文件选择框中,确定出所述Python实例化组件对应的python文件;如此,通过与该Python实例化组件的人机交互操作(如点击操作),则可在弹出的选择框内选取需要加载的python文件,而该文件主要用于解析所包含的类,以及setp_impl函数所包含的输入、输出参数数量等,因此,可作为配置项设置到前述Python实例化组件中。

[0065] 在调用得到对应的python文件后,即可基于此,来生成Sysplorer模型的Modelica文本信息;其中,举例可以但不限于采用如下步骤S31～S33,来得到Modelica文本信息。

[0066] S31.对所述python文件进行解析处理,以得到所述python文件中的输入参数、输出参数以及参数总数量,其中,所述输入参数和所述输出参数为建模参数;在本实施例中,可先对所述python文件进行解析处理,得到所述python文件中的所有python类(类是指一个函数包,可以包含函数(方法)和变量(属性),其定义了具有相同属性和方法的对象的集合,是面向对象编程的基础);然后,判断各python类中是否包含有预设函数(本实施例中,举例所述预设函数可以但不限于包括setp_impl函数);具体的,若各个python类中包含有setp_impl函数,则可获取各个预设函数的参数列表;最后,基于各个参数列表,则可确定出所述python文件中的输入参数、输出参数以及参数总数量;当然,若任一预设函数中未含有所述预设函数,则需要生成错误提示信息,并以弹窗形式将所述错误提示信息进行可视化展示。

[0067] 如此,经过前述步骤,则可完成python文件的解析,得到文件中各个参数名称(即输入参数和输出参数的名称);而后,则需要确定出参数的数据类型,以便后续基于数据类型,来生成Modelica文本信息;其中,前述数据类型的确定过程可以但不限于如下述步骤S32所示。

[0068] S32.确定出所述python文件中每个输入参数和每个输出参数的数据类型;在具体实施时,举例可以但不限于在响应于数据类型配置人机交互操作时,从数据类型选项库中确定出每个输入参数和每个输出参数的数据类型;可选的,数据类型的可选项可以但不限于包括:Real、Integer、Boolean、Npy_Int、Npy_Int、Npy_Bool、Python Object等数据类型。

[0069] 如此,在得到每个输入参数和每个输出参数的数据类型后,则可结合python文件,来生成Sysplorer模型的Modelica文本信息,其过程可以但不限于如下述步骤S33所示。

[0070] S33.基于所述python文件,以及所述python文件中每个输入参数的数据类型、每个输出参数的数据类型以及参数总数量,生成所述Sysplorer模型的Modelica文本信息;在本实施例中,是先获取所述python文件的文件路径和环境依赖路径,然后,基于所述文件路径、所述环境依赖路径、所述参数总数量、每个输入参数的数据类型以及每个输出参数的数据类型,生成所述Sysplorer模型的Modelica文本信息;具体的,则是将前述数据记录中Modelica文本中的各个字段中(即前述举例的inputDims、inputTypes、outputDims、outputTypes字段),从而在记录完成后,得到Modelica文本信息。

[0071] 由此,通过前述步骤S31～S33,则可利用python文件,来生成对应的Modelica文本信息;而后,基于该Modelica文本信息,则可直接构建出Modelica模型,其过程可以但不限于如下述步骤S4所示。

[0072] S4.利用所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型;在本实施例中,举例可以但不限于先获取Modelica仿真函数库;然后,基于所述Modelica仿真函数库和所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型。

[0073] 可选的,Modelica仿真函数库是一个外部函数库,其是基于C/C++语言开发的DLL动态运行库,用于支持Modelica语言在调用Python代码时对Python环境进行支持;其中,该函数库规定了函数的调用流程、数据的传输方式、输入输出端口数量计算等,并可使用指定版本的Python环境,加载py文件进行仿真计算工作。

[0074] 进一步的,该函数库包括初始化接口功能、数据交互接口功能和资源释放接口功能;其中,初始化接口用于初始化Python文件运行环境、内存资源申请,内存资源初始化,且在仿真计算过程中,被PythonInterface模型库constructor模块所调用;数据交互接口用于数据交互,即将模型当前的数据输入到python函数中去进行计算,并将结果重新反馈到模型中,在仿真计算过程中,其被PythonInterface模型库中exchangeData模块所调用;而资源释放接口则用于结束使用python运行环境,释放系统资源,且在仿真计算结束时,被PythonInterface模型库中destructor模块所调用。

[0075] 如此,则可依次调用外部函数库中的initPythonMemory、stepImpl、freePythonMemory函数,来完成模型仿真构建处理,从而得到Sysplorer模型对应的Modelica模型。

[0076] 由此通过前述步骤S11～S4所详细描述的基于Python模型库的Modelica建模方法,本发明通过建立Python-Modelica联合建模的方式,使得可在Modelica环境中直接使用python算法来完成建模,且整个过程由人工操作变为自动化运行,如此,不仅降低了出错概率,减少了模型开发人员的工作量,且还提升了建模效率;因此,非常适用于大规模应用与推广。

[0077] 如图2所示,本实施例第二方面提供了一种实现实施例第一方面中所述的基于Python模型库的Modelica建模方法的硬件装置,包括:

[0078] 库导入单元,用于获取PythonInterface模型库,并将所述PythonInterface模型库导入至建模工具内,其中,所述建模工具包括Sysplorer建模工具。

[0079] 实例化单元,用于基于所述建模工具,生成Sysplorer模型,并利用所述建模工具内的PythonInterface模型库,生成所述Sysplorer模型的Python实例化组件。

[0080] Modelica文本信息生成单元,用于获取所述Python实例化组件的python文件,并基于所述python文件,生成所述Sysplorer模型的Modelica文本信息。

[0081] 模型生成单元,用于利用所述Modelica文本信息,生成所述Sysplorer模型对应的Modelica模型。

[0082] 本实施例提供的装置的工作过程、工作细节和技术效果,可以参见实施例第一方面,于此不再赘述。

[0083] 如图3所示,本实施例第三方面提供了另一种基于Python模型库的Modelica建模装置,以装置为电子设备为例,包括:依次通信相连的存储器、处理器和收发器,其中,所述

存储器用于存储计算机程序,所述收发器用于收发消息,所述处理器用于读取所述计算机程序,执行如实施例第一方面所述的基于Python模型库的Modelica建模方法。

[0084] 具体举例的,所述存储器可以但不限于包括随机存取存储器(random access memory,RAM)、只读存储器(Read Only Memory,ROM)、闪存(Flash Memory)、先进先出存储器(First Input First Output,FIFO)和/或先进后出存储器(First In Last Out,FILO)等等;具体地,处理器可以包括一个或多个处理核心,比如4核心处理器、8核心处理器等。处理器可以采用DSP(Digital Signal Processing,数字信号处理)、FPGA(Field—Programmable Gate Array,现场可编程门阵列)、PLA(Programmable Logic Array,可编程逻辑阵列)中的至少一种硬件形式来实现,同时,处理器也可以包括主处理器和协处理器,主处理器是用于对在唤醒状态下的数据进行处理的处理器,也称CPU(Central Processing Unit,中央处理器);协处理器是用于对在待机状态下的数据进行处理的低功耗处理器。

[0085] 在一些实施例中,处理器可以在集成有GPU(Graphics Processing Unit,图像处理器),GPU用于负责显示屏所需要显示的内容的渲染和绘制,例如,所述处理器可以不限于采用型号为STM32F105系列的微处理器、精简指令集计算机(reduced instruction set computer,RISC)微处理器、X86等架构处理器或集成嵌入式神经网络处理器(neural-network processing units,NPU)的处理器;所述收发器可以但不限于为无线保真(WIFI)无线收发器、蓝牙无线收发器、通用分组无线服务技术(General Packet Radio Service,GPRS)无线收发器、紫蜂协议(基于IEEE802.15.4标准的低功耗局域网协议,ZigBee)无线收发器、3G收发器、4G收发器和/或5G收发器等。此外,所述装置还可以但不限于包括有电源模块、显示屏和其它必要的部件。

[0086] 本实施例提供的电子设备的工作过程、工作细节和技术效果,可以参见实施例第一方面,于此不再赘述。

[0087] 本实施例第四方面提供了一种存储包含有实施例第一方面所述的基于Python模型库的Modelica建模方法的指令的存储介质,即所述存储介质上存储有指令,当所述指令在计算机上运行时,执行如实施例第一方面所述的基于Python模型库的Modelica建模方法。

[0088] 其中,所述存储介质是指存储数据的载体,可以但不限于包括软盘、光盘、硬盘、闪存、优盘和/或记忆棒(Memory Stick)等,所述计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。

[0089] 本实施例提供的存储介质的工作过程、工作细节和技术效果,可以参见实施例第一方面,于此不再赘述。

[0090] 本实施例第五方面提供了一种包含指令的计算机程序产品,当所述指令在计算机上运行时,使所述计算机执行如实施例第一方面所述的基于Python模型库的Modelica建模方法,其中,所述计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。

[0091] 最后应说明的是:以上所述仅为本发明的优选实施例而已,并不用于限制本发明的保护范围。凡在本发明的精神和原则之内,所作的任何修改、等同替换、改进等,均应包含在本发明的保护范围之内。

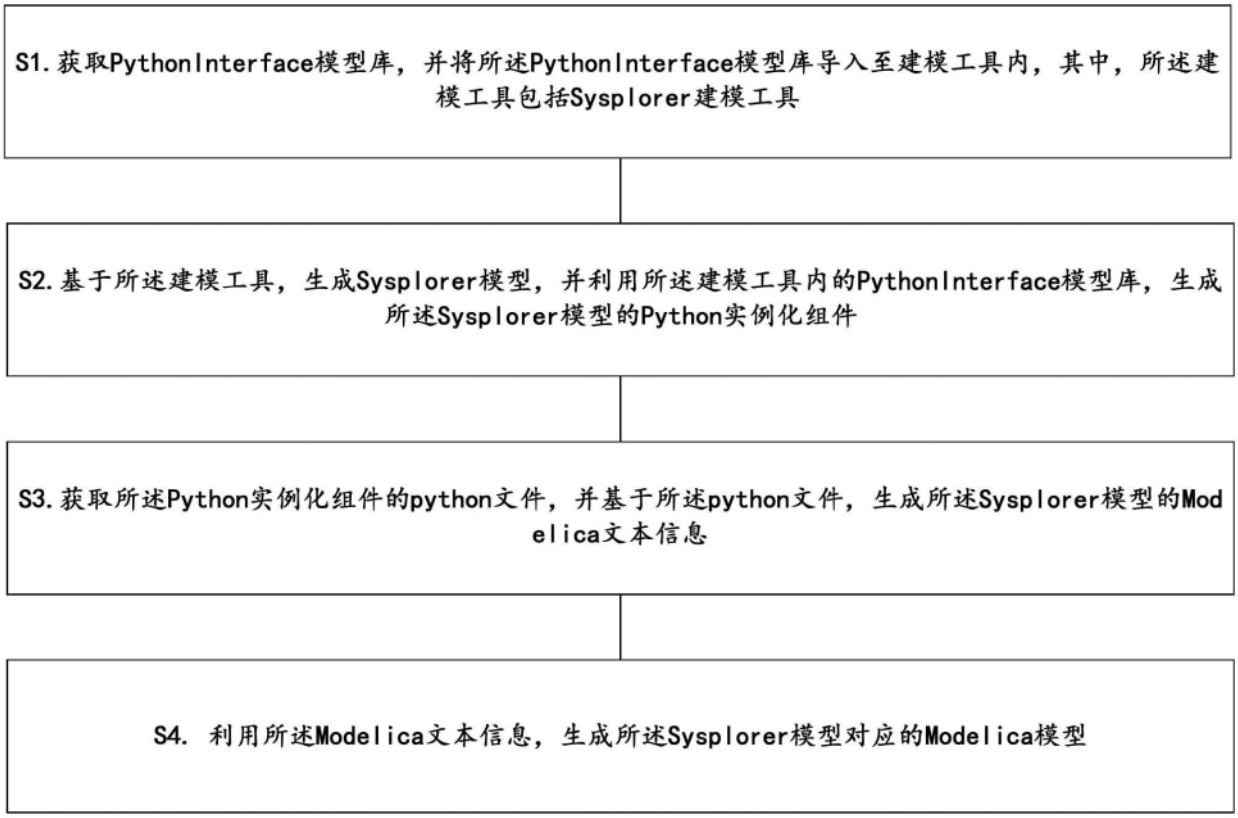
S1.获取PythonInterface模型库，并将所述PythonInterface模型库导入至建模工具内，其中，所述建模工具包括Sysplorer建模工具
S2.基于所述建模工具，生成Sysplorer模型，并利用所述建模工具内的PythonInterface模型库，生成所述Sysplorer模型的Python实例化组件
S3.获取所述Python实例化组件的python文件，并基于所述python文件，生成所述Sysplorer模型的Modelica文本信息
S4. 利用所述Modelica文本信息，生成所述Sysplorer模型对应的Modelica模型

图1

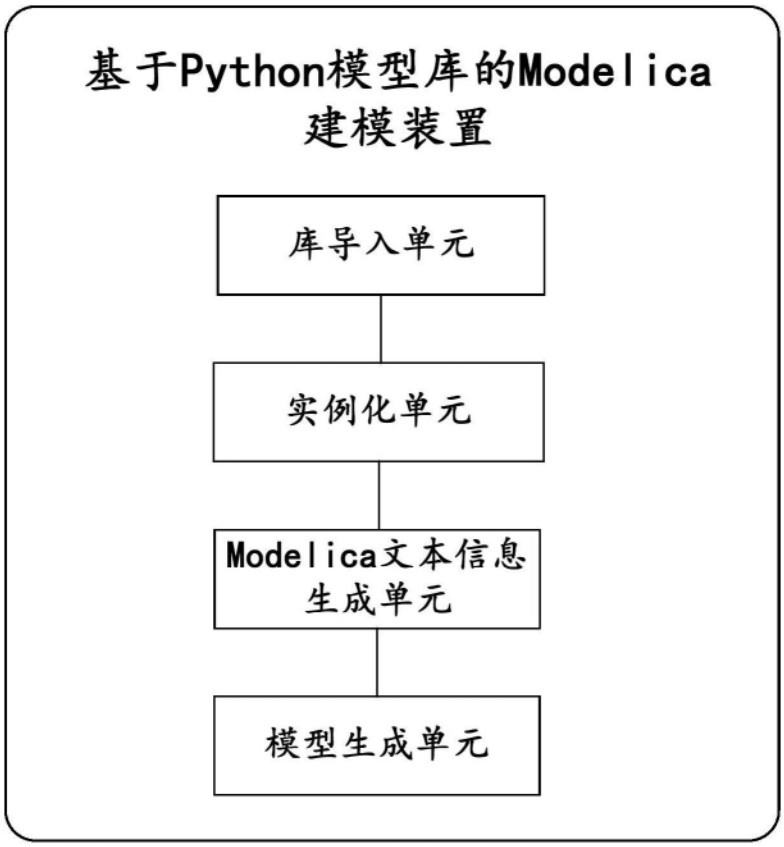
基于Python模型库的Modelica建模装置
库导入单元
实例化单元
Modelica文本信息生成单元
模型生成单元

图2

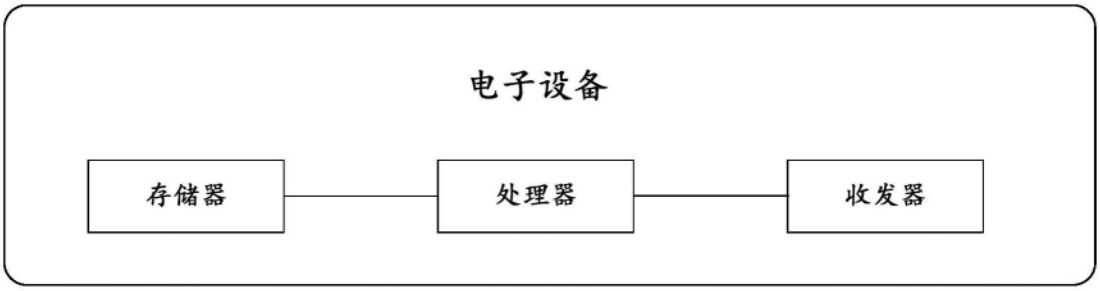
电子设备
存储器
处理器
收发器

图3