



SimCube. Automation 建立模型用户手册

Version 1.0

目 录

1.	可视化建立模型	1
1.1	仿函库	1
1.1.1	通过鼠标拖放添加仿函.....	1
1.1.2	仿函共有属性.....	1
1.2	可视化添加驱动仿函	3
1.2.1	添加方法.....	3
1.2.2	驱动仿函是什么.....	4
1.2.3	驱动仿函的类别.....	4
1.3	可视化建立基础仿函	13
1.3.1	添加方法.....	13
1.3.2	基础仿函是什么.....	14
1.3.3	基础仿函的类别.....	14
1.4	可视化添加参数仿函	22
1.4.1	添加方法.....	22
1.4.2	参数是什么.....	22
1.4.3	参数仿函的类别.....	23
1.5	可视化链接参数	31
1.5.1	链接参数是什么.....	31
1.5.2	链接参数的操作.....	31
2.	脚本建立模型	34
2.1	JavaScript 命名规则.....	34
2.2	脚本建立驱动仿函	35
2.2.1	装配仿函.....	35
2.2.2	序列仿函.....	35
2.2.3	并行仿函.....	35
2.2.4	分支仿函.....	36

2.2.5	总线仿函.....	37
2.2.6	循环仿函.....	38
2.3	脚本建立基础仿函.....	39
2.3.1	COM 仿函.....	39
2.3.2	仿函.....	40
2.3.3	脚本仿函.....	41
2.3.4	应用仿函.....	42
2.3.5	模型数据提取仿函.....	42
2.3.6	数据库仿函.....	43
2.4	脚本建立参数仿函.....	46
2.4.1	组.....	46
2.4.2	整数.....	47
2.4.3	双精度.....	47
2.4.4	布尔.....	48
2.4.5	字符串.....	48
2.4.6	文件.....	49
2.4.7	整数数组.....	49
2.4.8	双精度数组.....	50
2.4.9	布尔数组.....	51
2.4.10	字符串数组.....	52
2.4.11	表.....	53
2.5	脚本链接参数.....	56

1. 可视化建立模型

软件启动后的界面，如下图所示。



1.1 仿函数库

1.1.1 通过鼠标拖放添加仿函

仿函数库分为仿函数库、参数库，将仿函数添加到工作区的方法是：

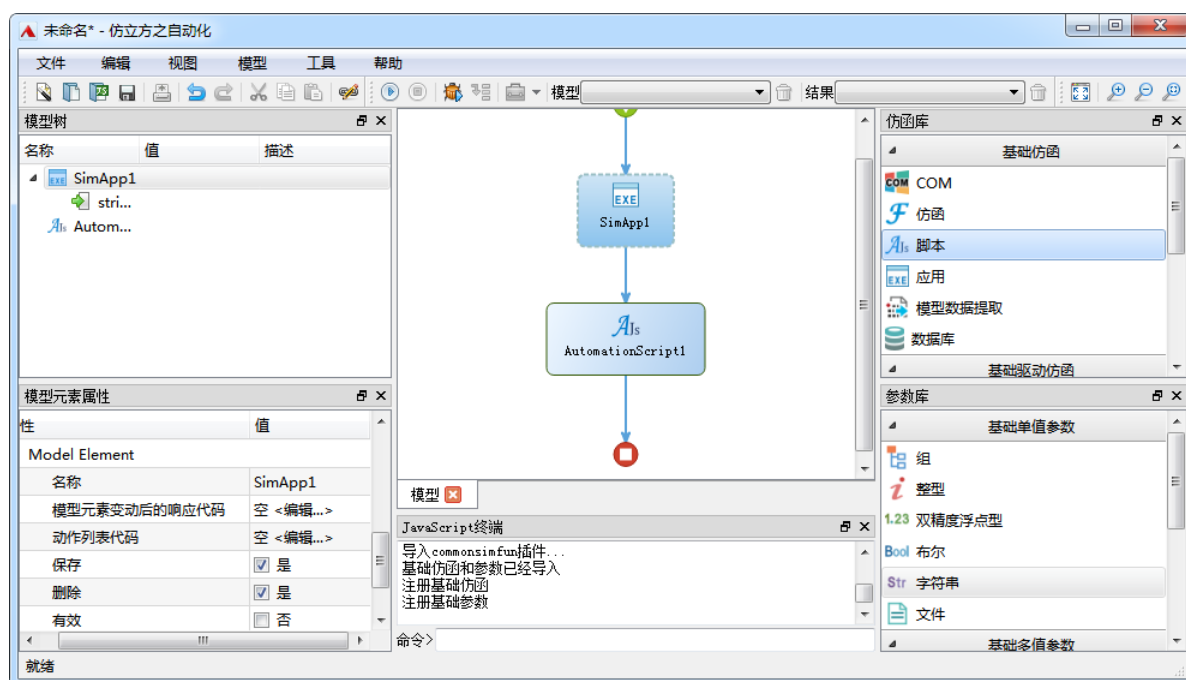
- 1) 先鼠标左键选择某个仿函，不放开；
- 2) 拖动仿函至工作区；
- 3) 在工作区合适的地方松开鼠标左键，仿函即可被添加进来。

1.1.2 仿函数共有属性

每个仿函数都会有如下属性，下面是对共有属性的说明。

■ 模型元素

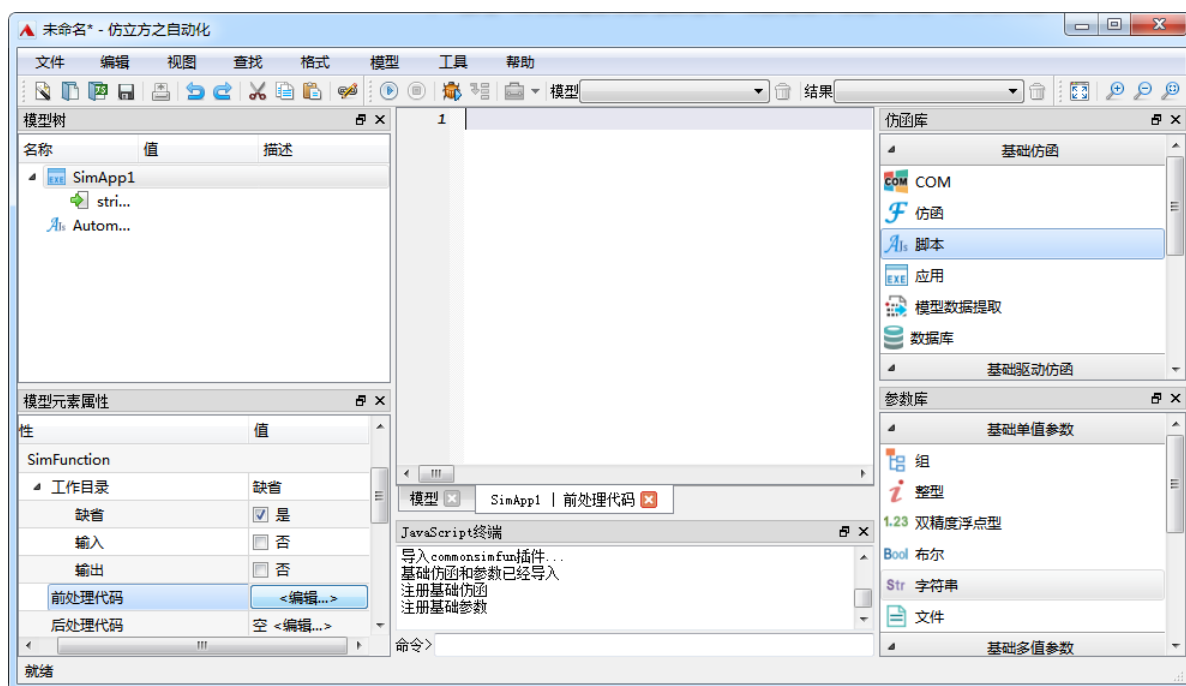
模型元素的共有属性，如下图所示。



- 名称：用户左键单击名称属性值，该控件变为可输入状态，用户可以修改仿函的名称；
- 模型元素变动后响应代码：当模型元素发生变动时，会发出信号，仿函接收到信号后，会执行相应的代码；
- 动作列表代码：用来执行 Action 列表的代码；
- 保存：用来设置该仿函是否保存到数据库中，默认选项是保存。
- 删除：用来设置该仿函是否可以被删除，默认是可删除的。
- 有效：用来设置该仿函是否有效，默认是有效的一个仿函。
- 描述：用户左键单击描述属性值，该控件变为可输入状态，用户可以输入该仿函的说明性文字；

■ 仿函

仿函的共有属性，如下图所示。



- 工作目录:用来定义工作目录的输入输出状态,如果工作目录是输入状态,勾选“输入”复选框;如果工作目录是输出状态,勾选“输出”复选框;如果同时为输入和输出状态,同时勾选“缺省”复选框;
- 前处理代码:在运行该仿函前可以执行一些代码。鼠标左键单击该属性值,在工作区会显示“前处理代码”对话框,用户可以在里面输入 Javascript 代码。
- 后处理代码:在运行该仿函后可以执行一些代码。鼠标左键单击该属性值,会弹出“后处理代码”对话框,与前处理代码类似。

1.2 可视化添加驱动仿函

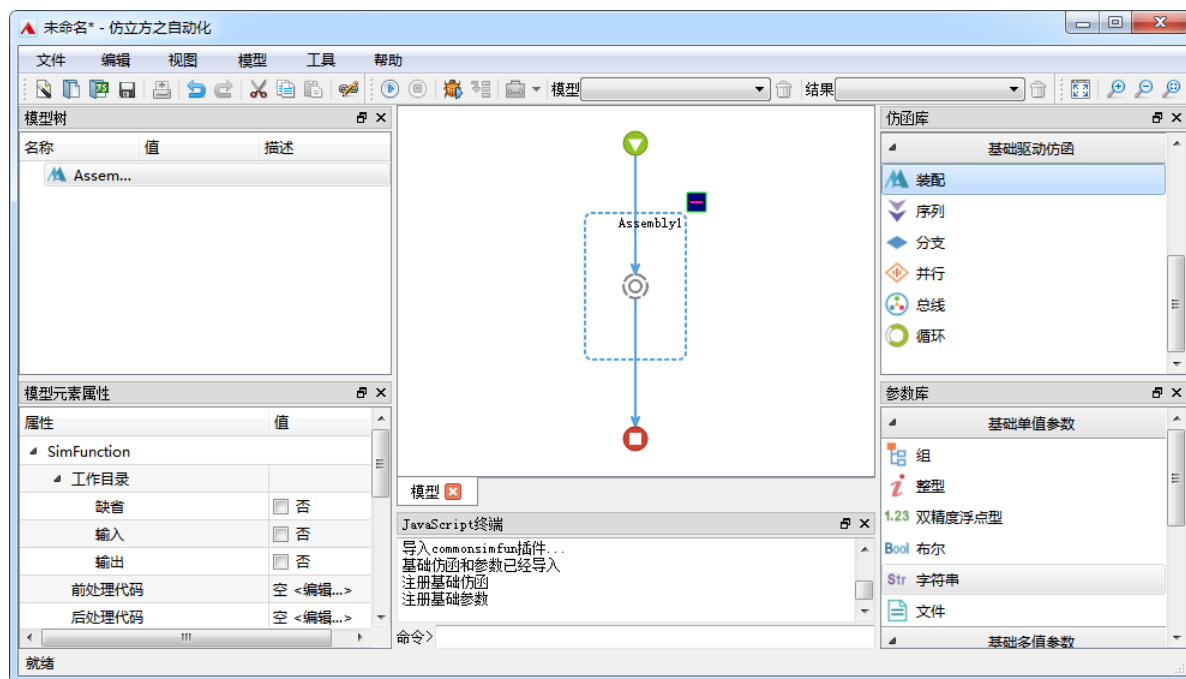
1.2.1 添加方法

以添加“装配”驱动仿函为例,说明添加的过程。在工作区中心位置有一个中心圆环,中心圆环表示可以添加新的仿函。拖拉“仿函数库|基础驱动仿函”中的“装配”驱动仿函到中心圆环,当拖拉至中心圆环附近时,中心圆环变为方形并高亮显示,如

下图所示。



此时松开鼠标左键，“装配”驱动仿函被添加至对应的位置上，如下图所示。



1.2.2 驱动仿函是什么

驱动仿函是复合仿函，可以在其内部添加其它的驱动仿函和模型元素。驱动仿函主要负责其内部仿函的执行序列，比如“序列”仿函表示按照先后顺序依次执行，“并行”仿函表示按照同时执行。

1.2.3 驱动仿函的类别

1.2.3.1 装配仿函

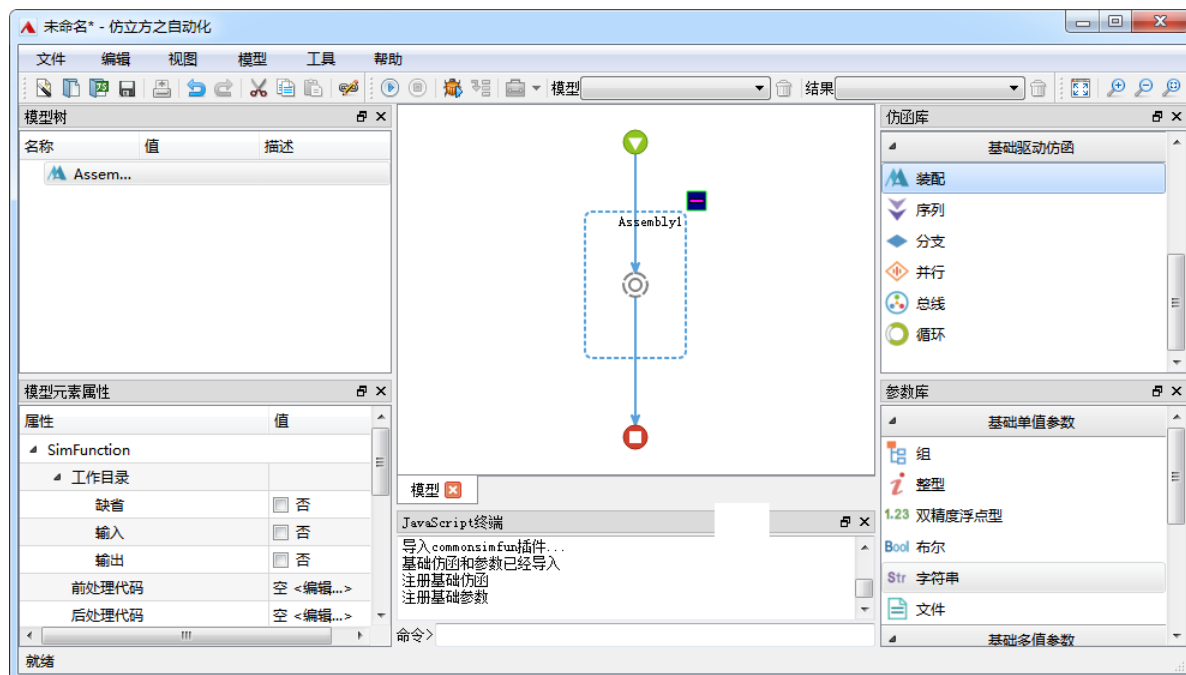
1.2.3.1.1 仿函功能

“装配”相当于模型的根节点，在“装配”驱动仿函中可以添加新的驱动仿函和模型仿函。当 SimCube.Automation 以“新建模型”的方式打开，新建的模型默认创建

了一个“装配”驱动仿函。

1.2.3.1.2 仿函的属性

在模型树或工作区中点击“装配”驱动仿函，在左下方的“模型元素属性”窗口中会列出该仿函的属性及属性值，用户可以进行设定，如下图所示。



该仿函只具有共有属性，详细内容请参考 1.1.2 节内容。

1.2.3.2 序列仿函

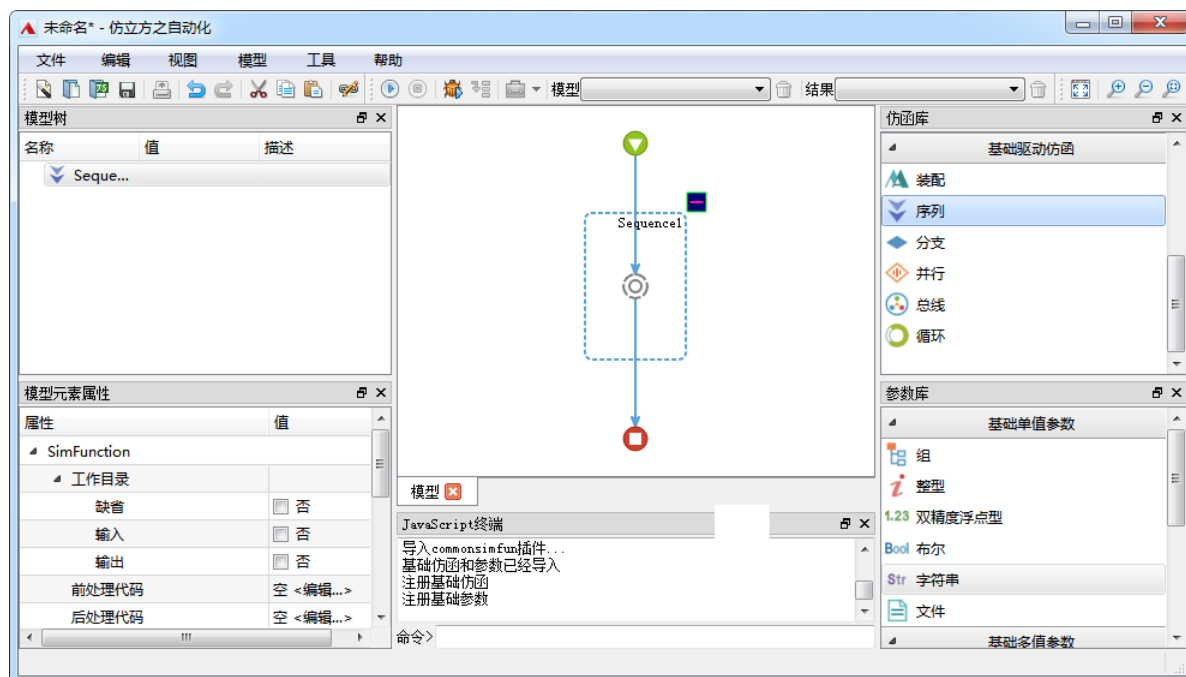
1.2.3.2.1 仿函功能

“序列”表示按先后序列依次执行内部的仿函，在“序列”仿函中可以添加新的驱动仿函和模型仿函。当 SimCube.Automation 以“新建模型”的方式打开，新建的模型默认执行方式为“序列”仿函。

1.2.3.2.2 仿函的属性

在模型树或工作区中点击“序列”仿函，在左下方的“模型元素属性”窗口中会

列出该仿函数的属性及属性值，用户可以进行设定，如下图所示。



该仿函数只具有共有属性，详细内容请参考 1.1.2 节内容。

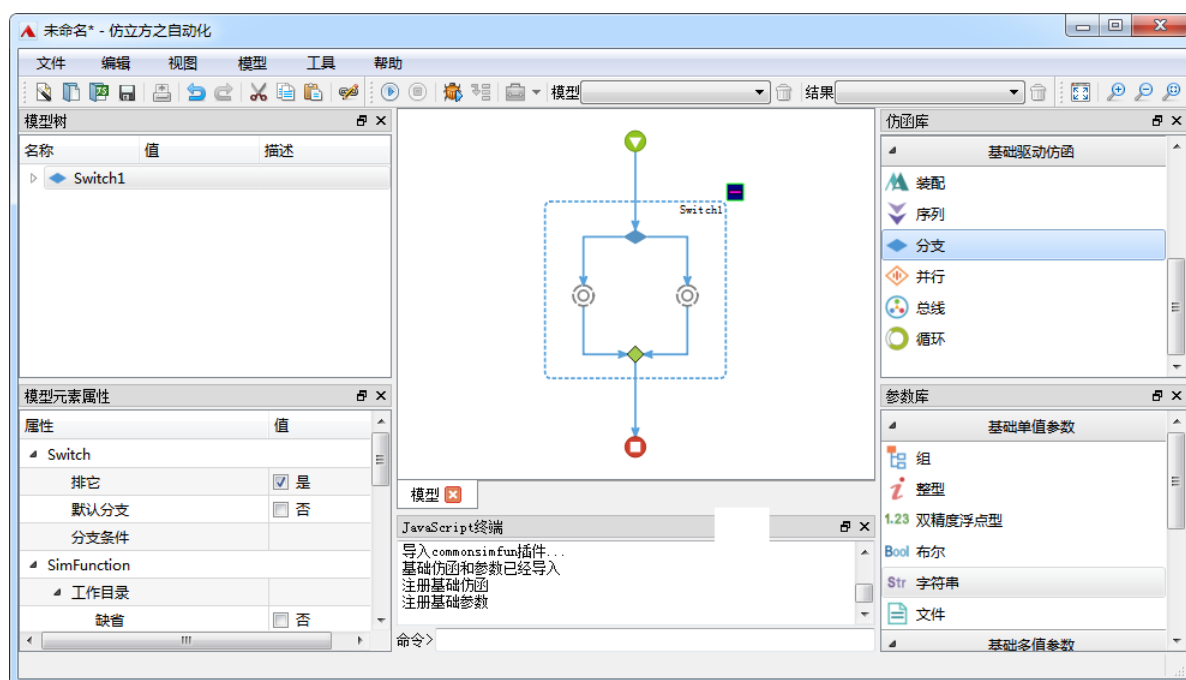
1.2.3.3 分支仿函数

1.2.3.3.1 仿函数功能

“分支”表示按照某种条件执行某个分支上的仿函数，在“分支”仿函数中可以添加新的驱动仿函数和模型仿函数。

1.2.3.3.2 仿函数的属性

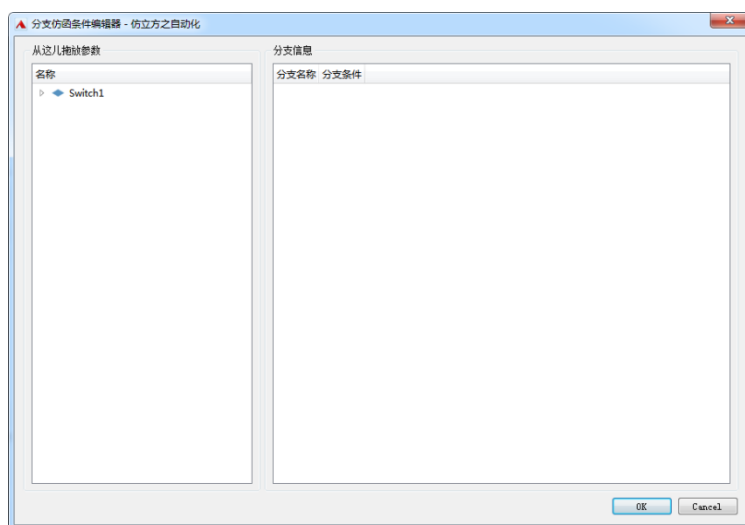
在模型树或工作区中点击“分支”仿函数，在左下方的“模型元素属性”窗口中会列出该仿函数的属性及属性值，用户可以进行设定，如下图所示。



除了共有属性，详细内容请参考 1.1.2 节内容，还有自己的属性。

■ Switch

- 排它：每次只运行满足条件的分支，默认勾选。
- 默认分支：执行时最先执行的分支，默认不勾选。
- 分支条件：用来设置分支的条件，点击“值”栏，会弹出设置对话框，如下图所示。



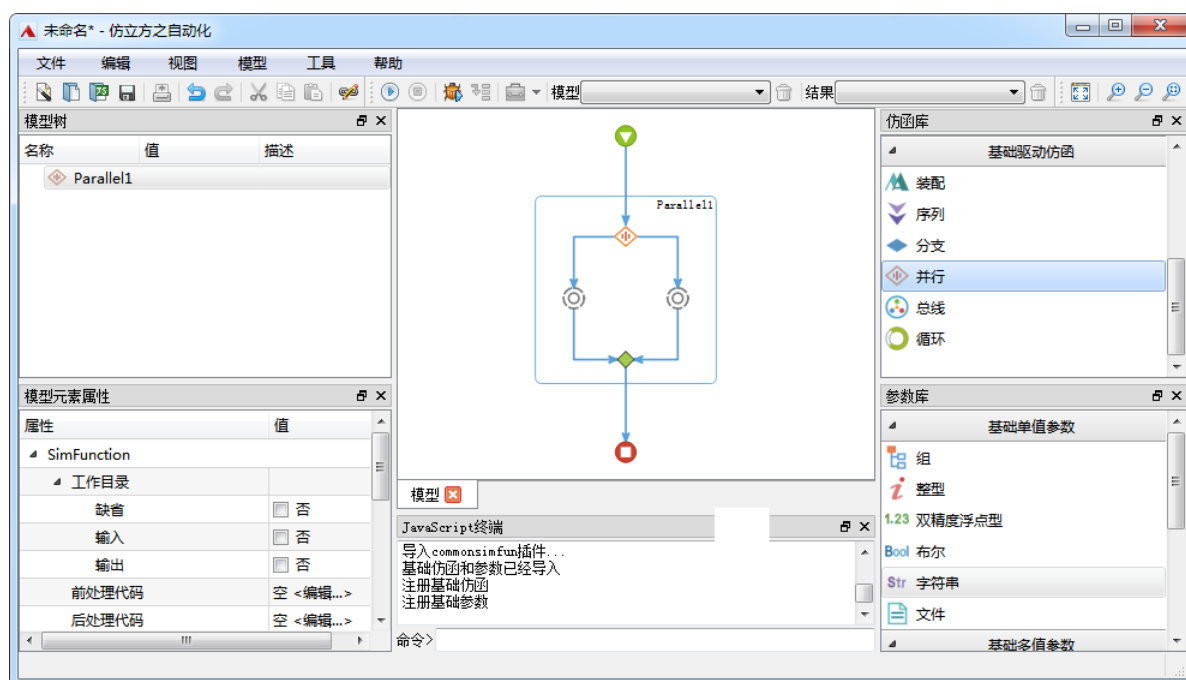
1.2.3.4 并行仿函

1.2.3.4.1 仿函功能

“并行”表示按并行依次执行内部的仿函，在“并行”仿函中可以添加新的驱动仿函和模型仿函。新建“并行”仿函默认只建立两个分支，当两个分支上都有仿函时，可以将仿函添加到第三个分支上。

1.2.3.4.2 仿函的属性

在模型树或工作区中点击“并行”仿函，在左下方的“模型元素属性”窗口中会列出该仿函的属性及属性值，用户可以进行设定，如下图所示。



该仿函只具有共有属性，详细内容请参考 1.1.2 节内容。

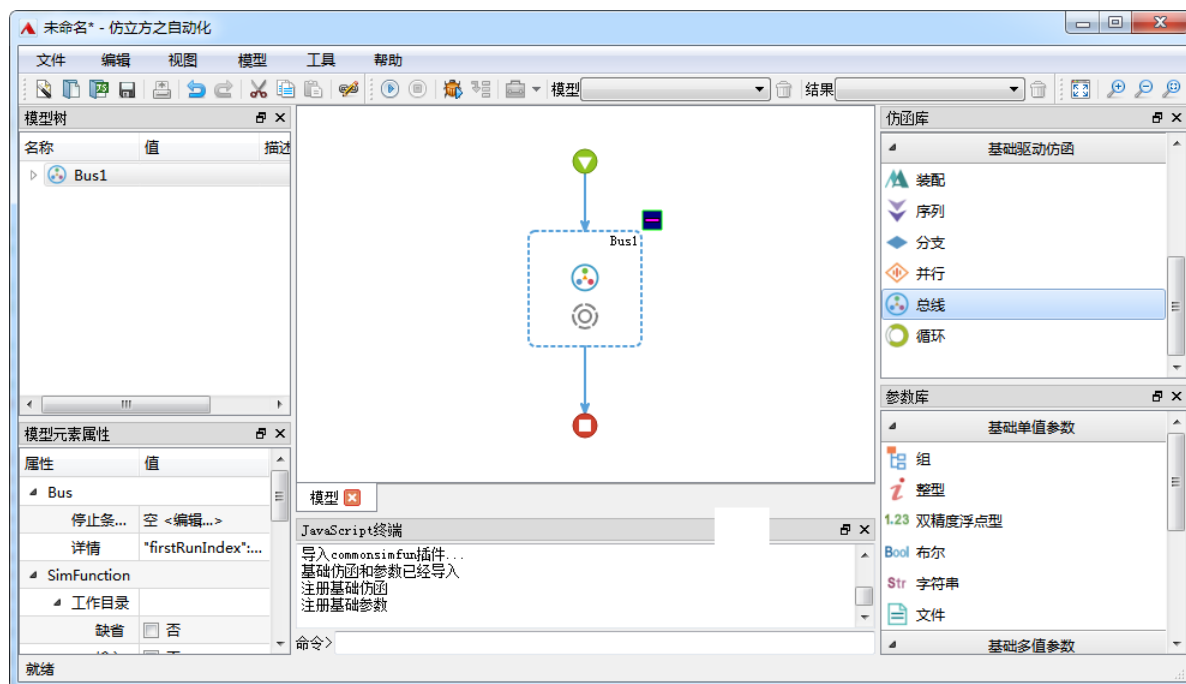
1.2.3.5 总线仿函

1.2.3.5.1 仿函功能

“总线”表示按某些条件来循环执行内部的仿函，当某个仿函条件满足，就执行；否则不执行。在“总线”仿函中可以添加新的驱动仿函和模型仿函。

1.2.3.5.2 仿函的属性

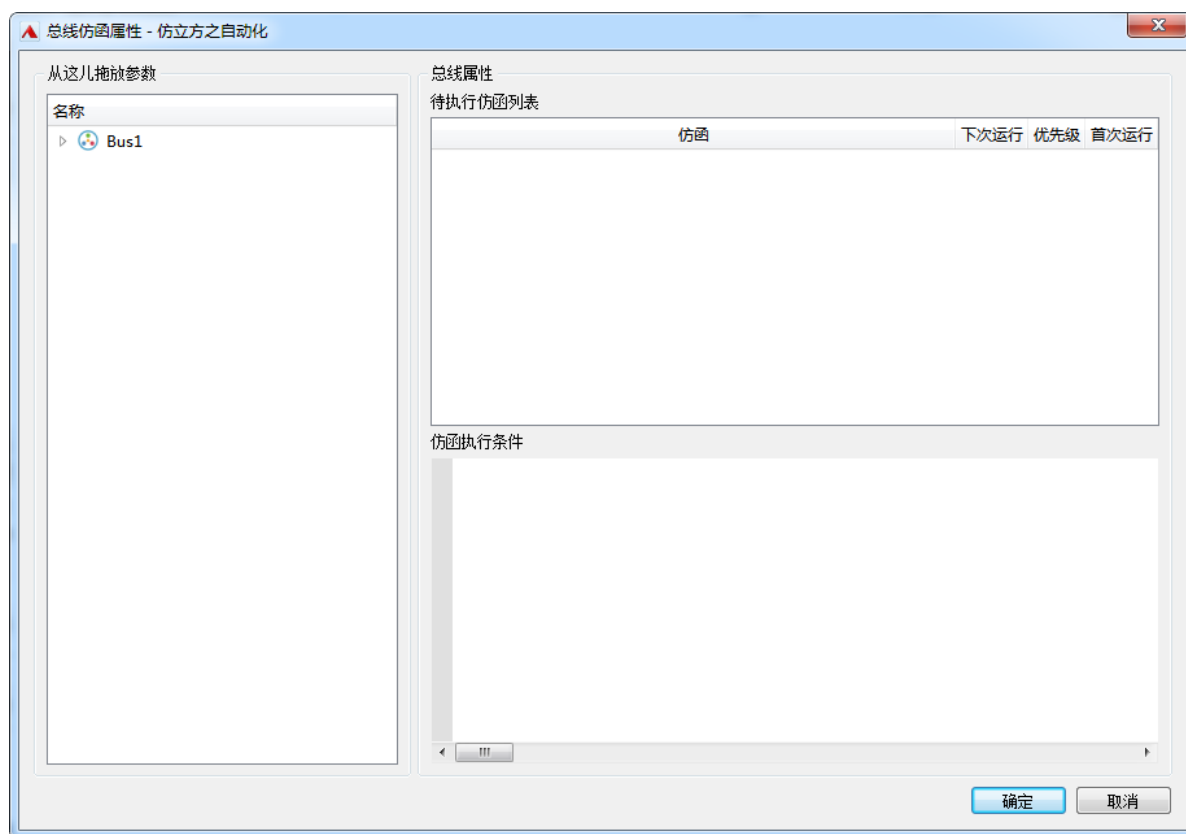
在模型树或工作区中点击“总线”仿函，在左下方的“模型元素属性”窗口中会列出该仿函的属性及属性值，用户可以进行设定，如下图所示。



除了共有属性，详细内容请参考 1.1.2 节内容，还有自己的属性。

■ Bus

- 停止条件：用户可以指定总线仿函数的停止运行条件；
- 详情：用来显示总线仿函数的详细设定信息，点击“值”栏中的编辑，弹出循环仿函数属性，以 For 循环为例，弹出的对话框如下图所示。



首次运行：整个 Bus 仿函运行时，第一个被执行的仿函；

优先级：如果 Bus 内部有多个仿函满足条件，只有优先级最高的仿函被执行；

下次运行：当前仿函运行完毕后，无条件运行设置的仿函；

仿函：列出 Bus 仿函内部的所有的仿函。

1.2.3.6 循环仿函

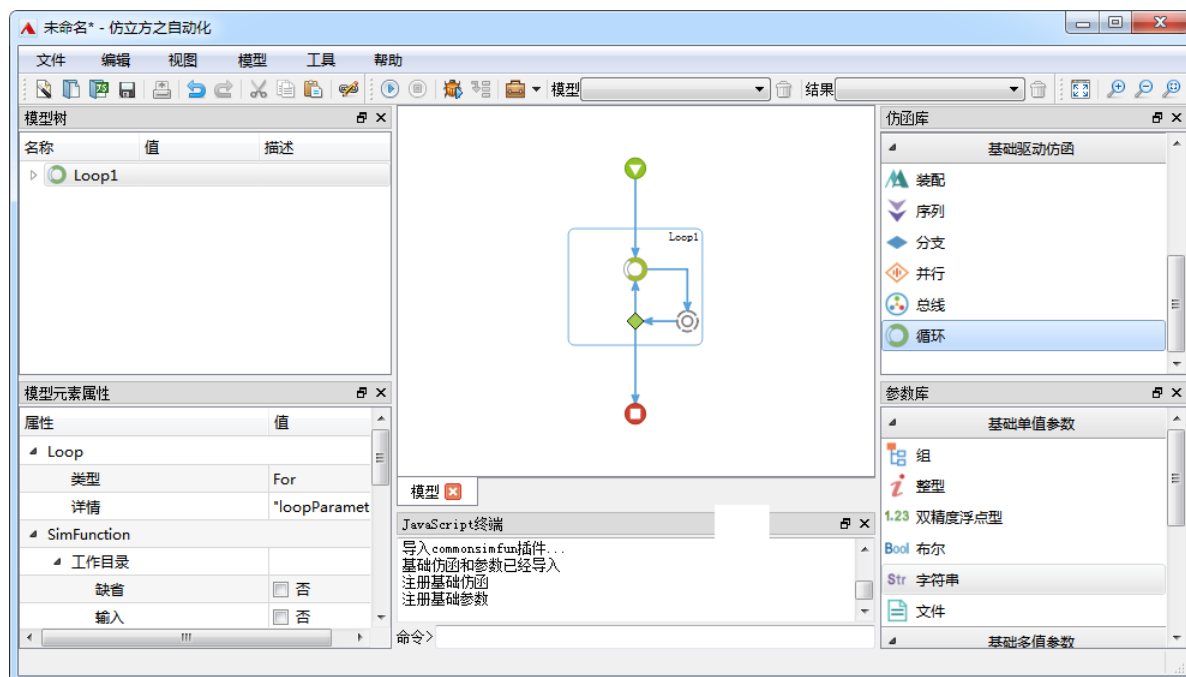
1.2.3.6.1 仿函功能

“循环”表示循环执行内部的仿函若干次，在“循环”仿函中可以添加新的驱动仿函和模型仿函。“循环”仿函内部会有一个变量值 counter，表示循环执行的次数。

1.2.3.6.2 仿函的属性

在模型树或工作区中点击“循环”仿函，在左下方的“模型元素属性”窗口中会

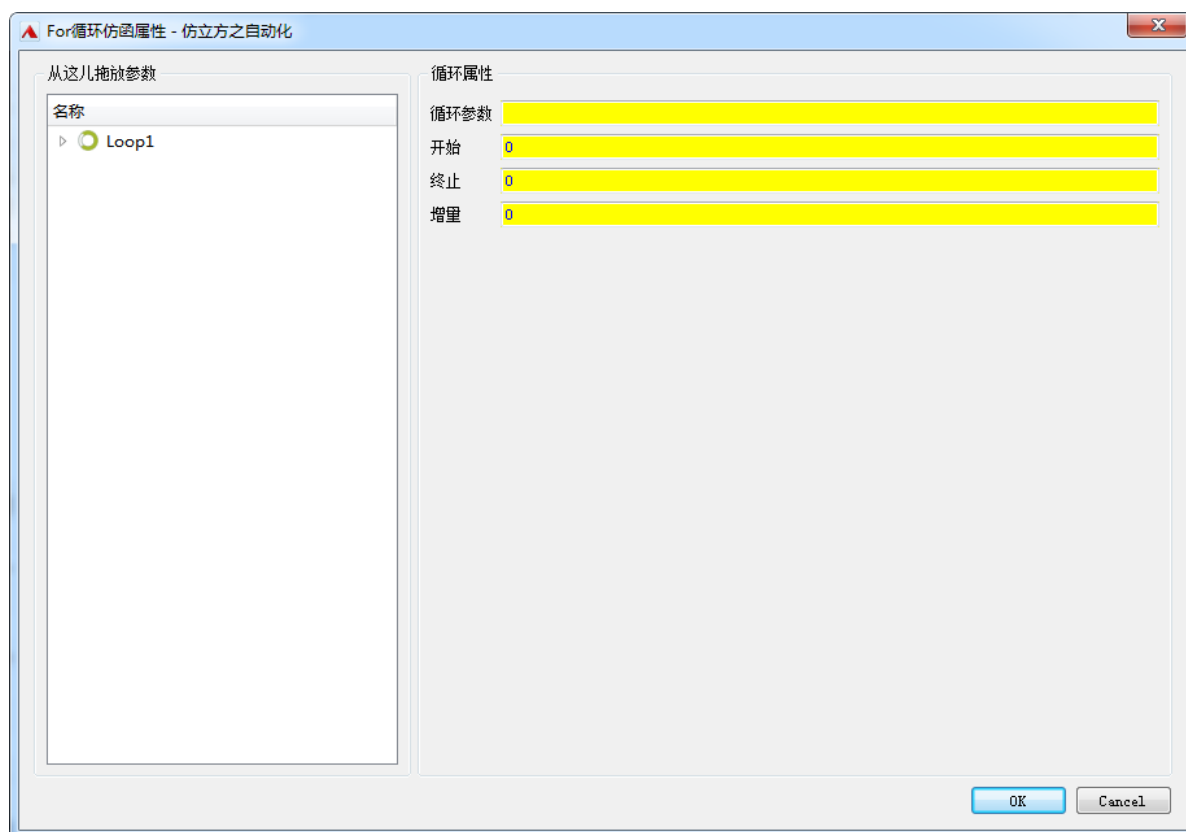
列出该仿函的属性及属性值，用户可以进行设定，如下图所示。



除了共有属性，详细内容请参考 1.1.2 节内容，还有自己的属性。

■ Loop

- 类型：用户可以指定循环的类型，Loop 支持的类型有三类，分别是 For、Repeat、RepeatUtil，默认是 For；
- 详情：用来显示指定类型循环的详细设定信息，点击“值”栏中的编辑，弹出循环仿函属性，以 For 循环为例，弹出的对话框如下图所示。



循环参数：用来设定进行循环的参数；

开始：循环的开始数值，默认是 0；

终止：循环的终止数值，默认是 0；

增量：循环的步长，默认是 0；

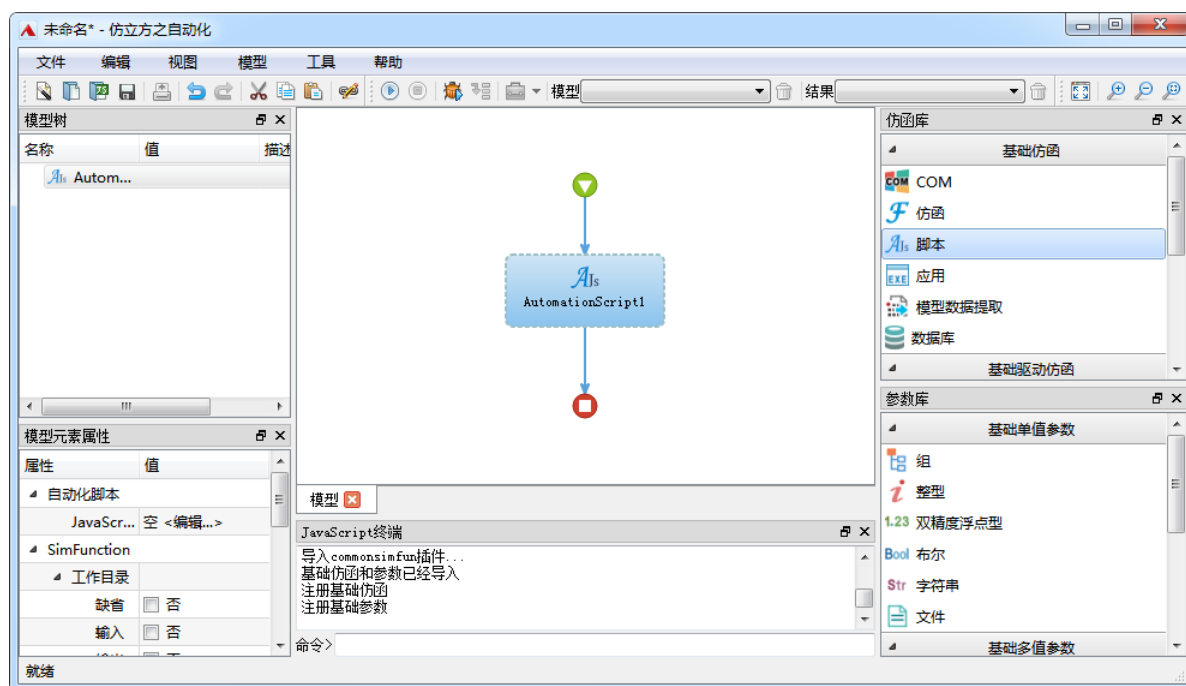
1.3 可视化建立基础仿函

1.3.1 添加方法

以添加“脚本”仿函为例，说明添加的过程。在工作区中心位置有一个中心圆环，中心圆环表示可以添加新的仿函。拖拉“基础仿函”库中的“脚本”驱动仿函到中心圆环，当拖拉至中心圆环附近时，中心圆环变为方形并高亮显示，如下图所示。



此时松开鼠标左键，“脚本”驱动仿函被添加至对应的位置上，如下图所示。



1.3.2 基础仿函数是什么

基础仿函数是完成特定功能的仿函数，与驱动仿函数的区别在于在其内部不可以添加其它的驱动仿函数和基础仿函数。

1.3.3 基础仿函数的类别

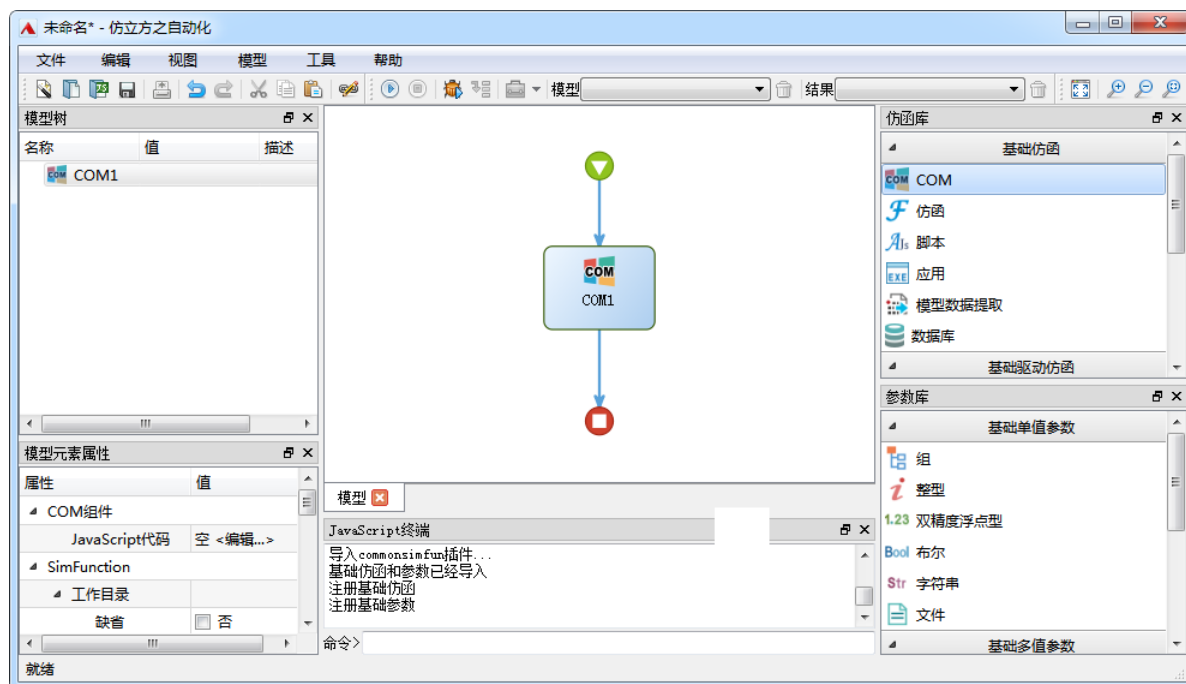
1.3.3.1 COM 仿函数

1.3.3.1.1 仿函数功能

“COM”仿函数可以直接调用 COM 组件中的函数，供基础仿函数或流程仿函数使用。

1.3.3.1.2 仿函数的属性

在模型树或工作区中点击“COM”仿函数，在左下方的“模型元素属性”窗口中会列出该仿函数的属性及属性值，用户可以进行设定，如下图所示。



除了共有属性，详细内容请参考 1.1.2 节内容，还有自己的属性。

■ COM 组件

JavaScript Code: 鼠标左键单击该属性值，会在工作区显示“编辑 Javascript 代码”界面，用户可以在里面输入 Javascript 代码。

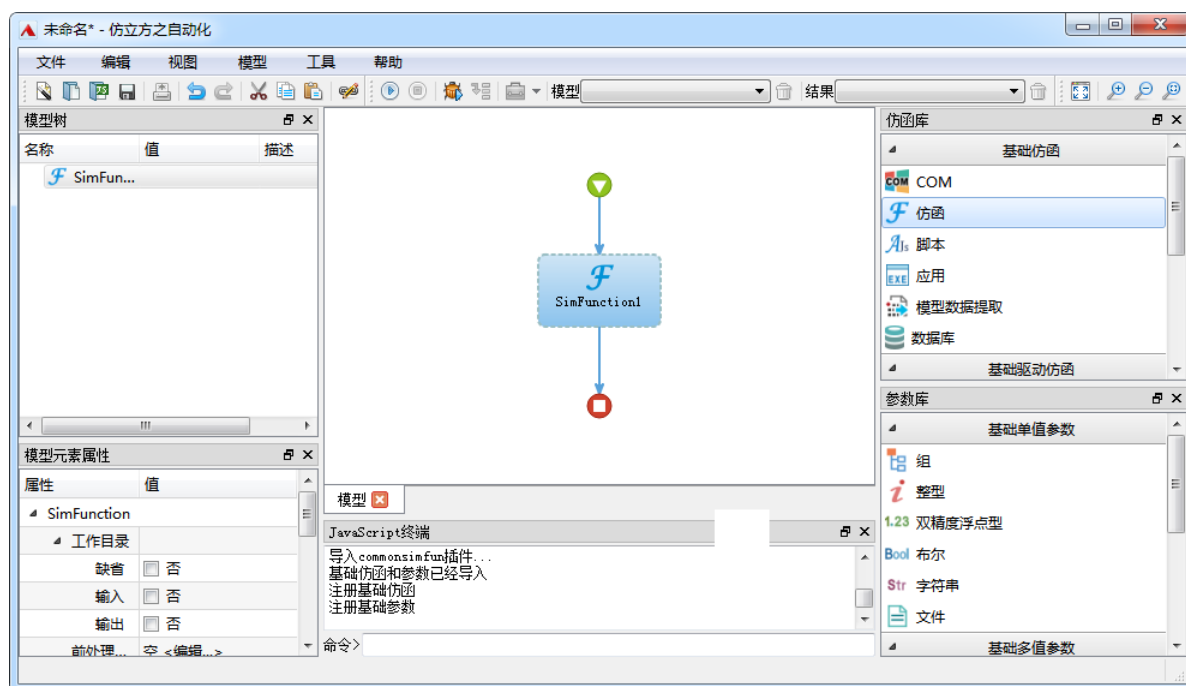
1.3.3.2 仿函

1.3.3.2.1 仿函功能

“仿函”没有具体的功能，只是起到占位符的作用。

1.3.3.2.2 仿函的属性

在模型树或工作区中点击“仿函”仿函，在左下方的“模型元素属性”窗口中会列出该仿函的属性及属性值，用户可以进行设定，如下图所示。



仿函中只有共有属性，详细内容请参考 1.1.2 节内容。

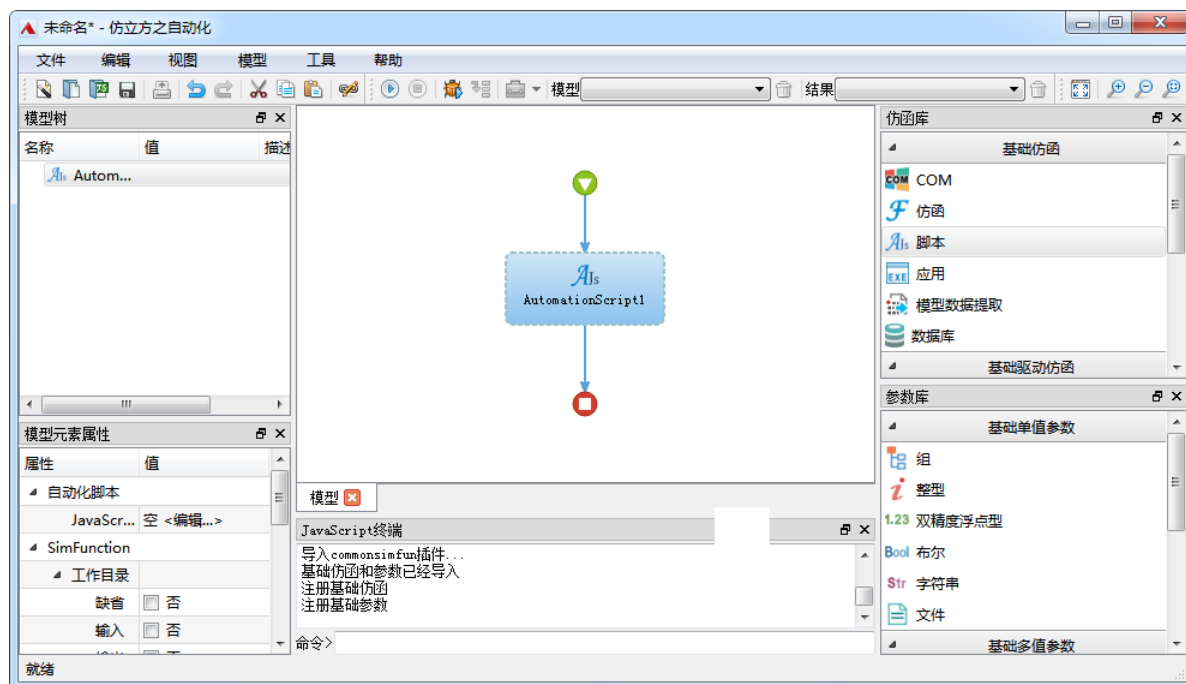
1.3.3.3 脚本仿函

1.3.3.3.1 仿函功能

“脚本”可以完成定制编写的 Javascript 代码。Javascript 是通用的脚本语言，功能强大，使用简便，用户可以方便地在 SimCube.Automation 中使用 Javascript 脚本来定制某项功能。

1.3.3.3.2 仿函的属性

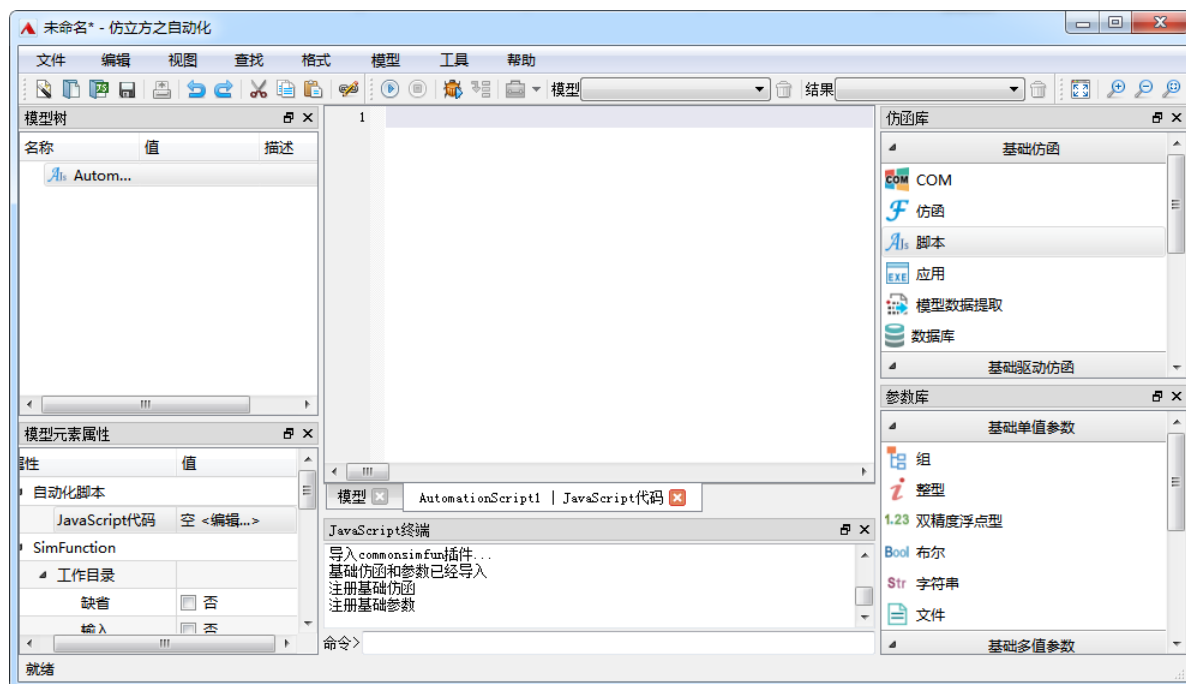
在模型树或工作区中点击“脚本”仿函，在左下方的“模型元素属性”窗口中会列出该仿函的属性及属性值，用户可以进行设定，如下图所示。



除了共有属性，详细内容请参考 1.1.2 节内容，还有自己的属性。

■ 自动化脚本

- Javascript Code: 鼠标左键单击该属性值，会在工作区显示“编辑 Javascript 代码”界面，用户可以在里面输入 Javascript 代码，如下图所示。



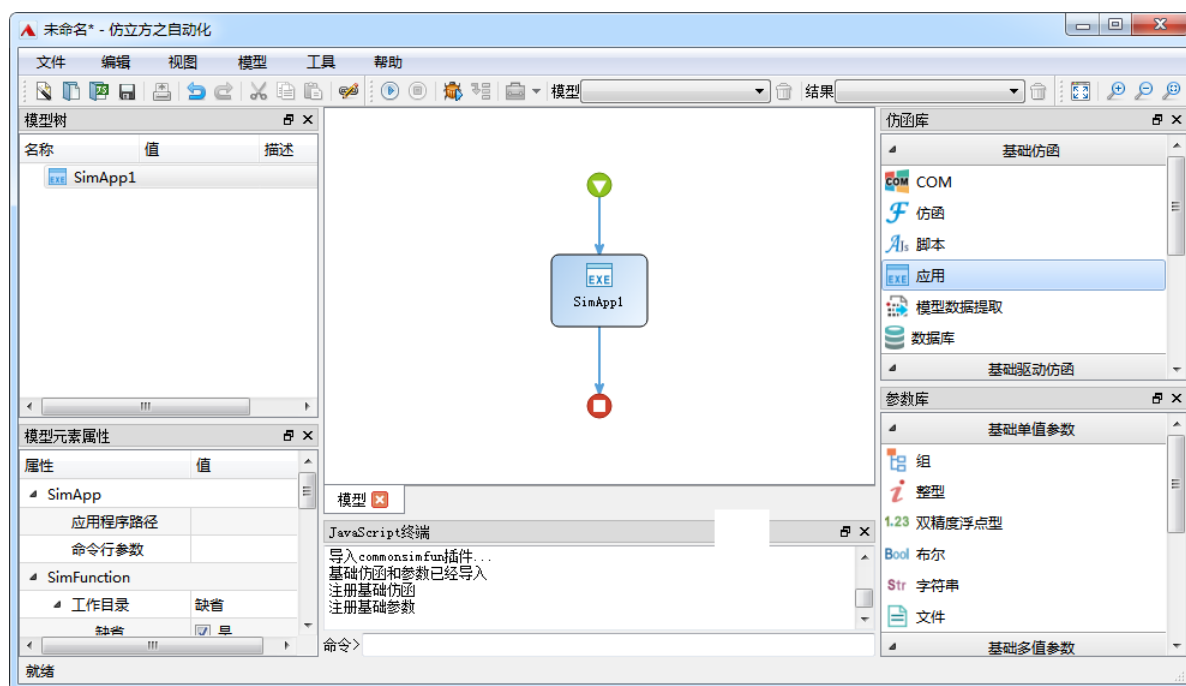
1.3.3.4应用仿函

1.3.3.4.1仿函功能

“应用”仿函主要用来封装用户编写的 HomeIn 程序，并调用执行，也可以调用商业软件。

1.3.3.4.2仿函的属性

在模型树或工作区中点击“应用”仿函，在左下方的“模型元素属性”窗口中会列出该仿函的属性及属性值，用户可以进行设定，如下图所示。



除了共有属性，详细内容请参考 1.1.2 节内容，还有自己的属性。

■ SimApp

- 应用程序路径：鼠标左键单击该属性值，点击“编辑”，弹出浏览 exe 文件的对话框，用户选择 exe 文件后，点击“确定”，exe 程序的绝对路径将返回到文本框内；
- 命令行参数：指定 exe 程序运行时所需要的参数。

1.3.3.5 模型数据提取仿函

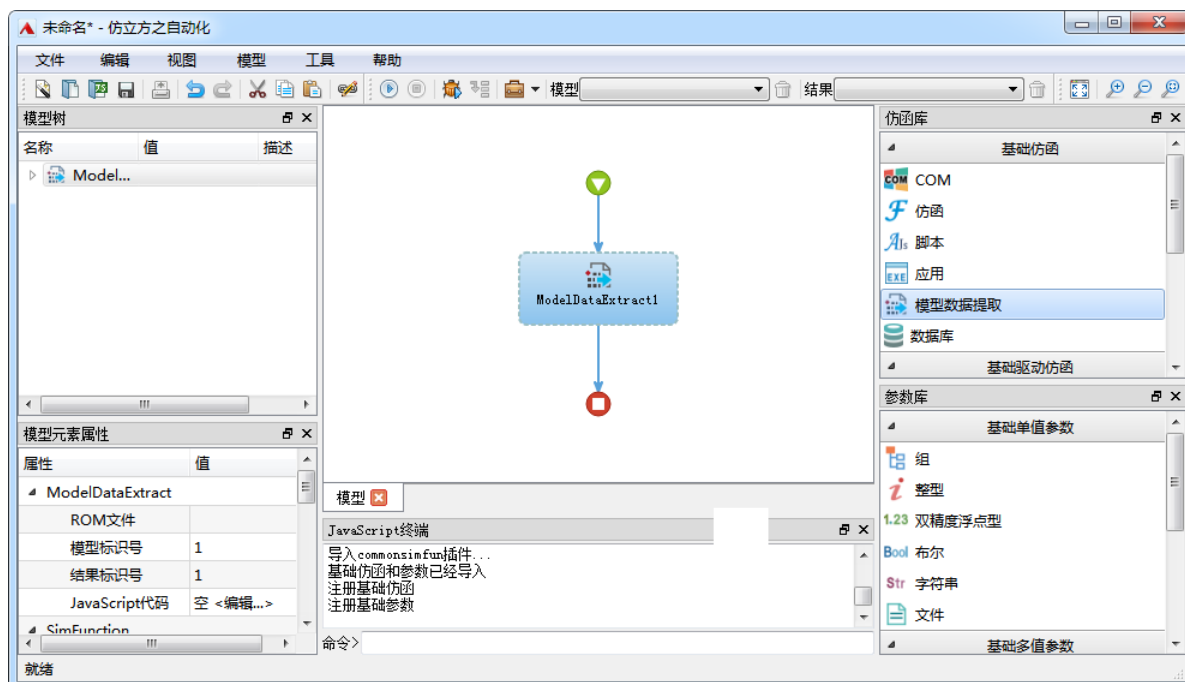
1.3.3.5.1 仿函功能

“模型数据提取”仿函可以从 rom 文件中提取出数据，供基础仿函或流程仿函使用。

1.3.3.5.2 仿函的属性

在模型树或工作区中点击“模型数据提取”仿函，在左下方的“模型元素属性”

窗口中会列出该仿函数的属性及属性值，用户可以进行设定，如下图所示。



除了共有属性，详细内容请参考 1.1.2 节内容，还有自己的属性。

■ ModelDataExtract

- ROM 文件：鼠标左键单击该属性值，点击“编辑”，弹出浏览 ROM 文件的对话框，用户选择 ROM 文件后，点击“确定”，ROM 文件的绝对路径将返回到文本框内；
- 模型标识号：每一次模型改动都会存为一个新版本，在此指定模型的标识号；
- 结果标识号：每一次模型运算都会产生一个新版本结果，在此指定结果的标识号；
- Javascript Code：鼠标左键单击该属性值，会在工作区显示“编辑 Javascript 代码”界面，用户可以在里面输入 Javascript 代码。

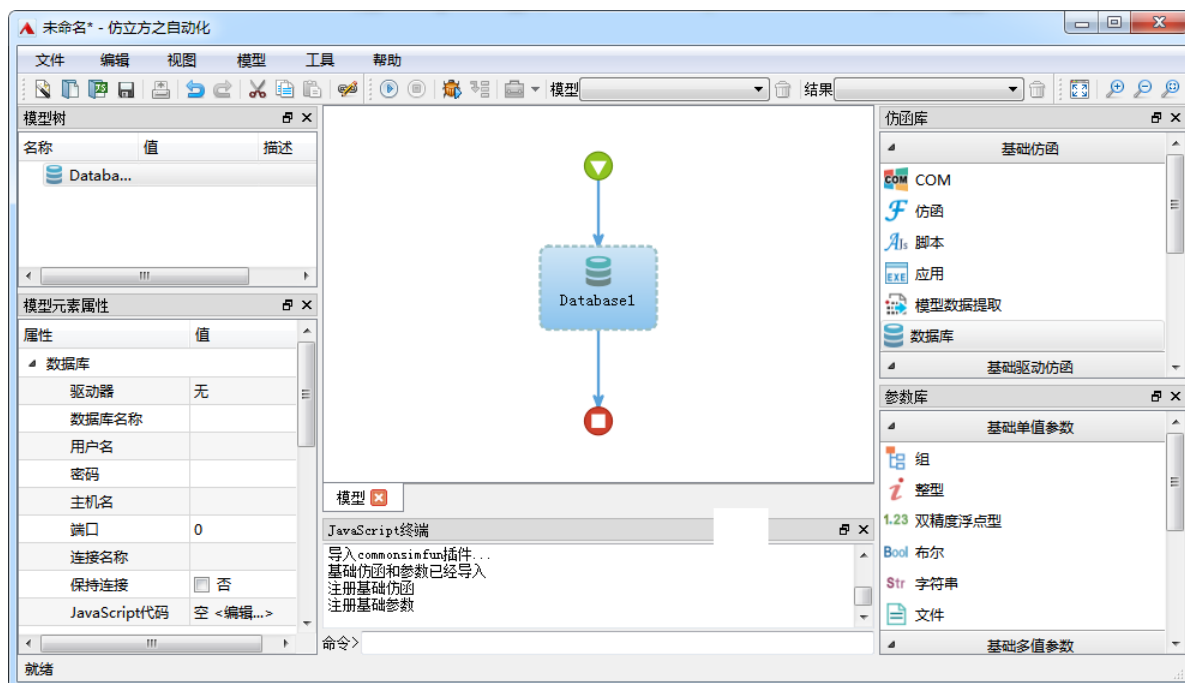
1.3.3.6 数据库仿函

1.3.3.6.1 仿函功能

“数据库”仿函可以从数据库文件中提取出数据，供基础仿函或流程仿函使用。

1.3.3.6.2 仿函的属性

在模型树或工作区中点击“数据库”仿函，在左下方的“模型元素属性”窗口中会列出该仿函的属性及属性值，用户可以进行设定，如下图所示。



除了共有属性，详细内容请参考 1.1.2 节内容，还有自己的属性。

■ 数据库

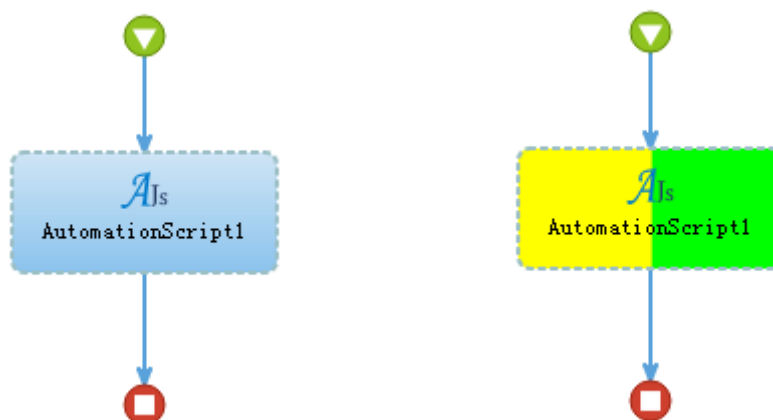
- 驱动器：连接数据库所需要的驱动程序，Automation 支持的数据库驱动器包括 SQLite、MySQL、MySQL 3、ODBC、ODBC 3、PostgreSQL、PostgreSQL 7 等。
- 数据库名称：设定数据库的名称；
- 用户名：设置登陆数据库的用户名；

- 密码：设置登陆数据库的密码；
- 主机名：设置数据库所在的主机名；
- 端口：设置数据库所在的主机的端口；
- 连接名称：设置与数据库连接的名称；
- 保持连接：设置是否一直保持连接；
- Javascript Code：鼠标左键单击该属性值，会在工作区显示“编辑 Javascript 代码”界面，用户可以在里面输入 Javascript 代码。

1.4 可视化添加参数仿函

1.4.1 添加方法

拖拉“基础单值”库中的“整数”驱动仿函到基础仿函上时，仿函高亮显示，并且分成两种颜色，左边是黄色，右边是绿色，如下图所示。



黄色代表输入项，绿色代表输出项。用户选择不同的位置，松开鼠标左键，数据参数即可被添加至仿函中。

1.4.2 参数是什么

每一个仿函要有输入、输出参数，仿函才会有存在的意义。目前支持的参数类型如下表所示。

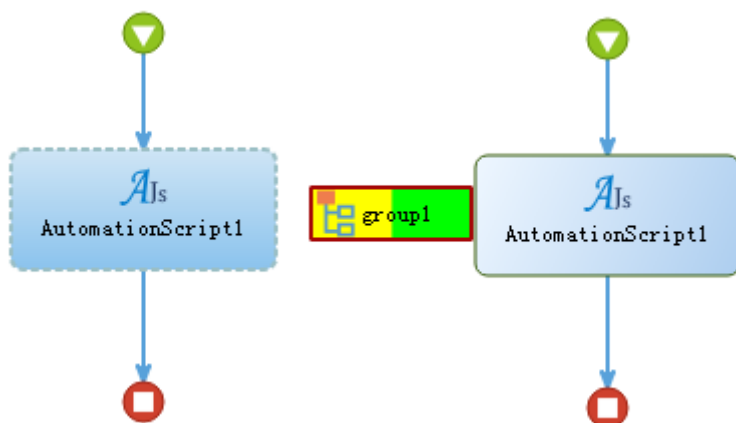
序号	类型
1	组
2	整型
3	双精度浮点类型
4	布尔
5	字符串
6	文件
7	整型数组
8	双精度数组
9	布尔数组
10	字符串数组
11	表
12	复合型

1.4.3 参数仿函的类别

1.4.3.1 组

组可以将不同类型的参数打包成集合，组下面不能包含组，但可以包含其它的各种类型的参数。

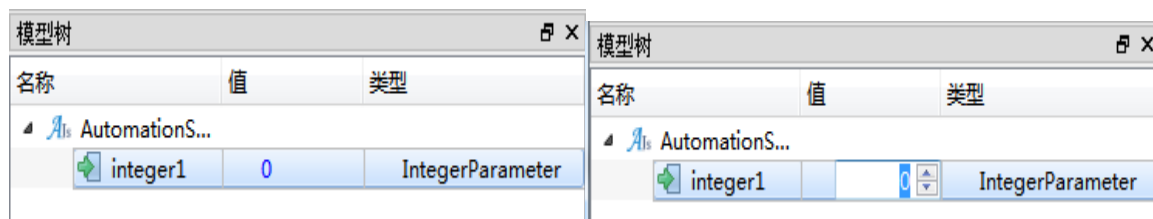
组仿函的添加方法，首先拖放一个组仿函到程序中，然后再拖放其它类型参数到组仿函上面，如下图所示。这时会显示出组仿函，然后将参数放于组仿函的输入或输出位置，松开即可。



1.4.3.2 整数

整数类型只接受整数类型的数值，用户不能输入非数字的内容，比如“。abc\d”等字符。

修改数值的方法如下，在模型树上，选择对应的参数，点击“值”栏，鼠标点入后，界面变为如下图所示。

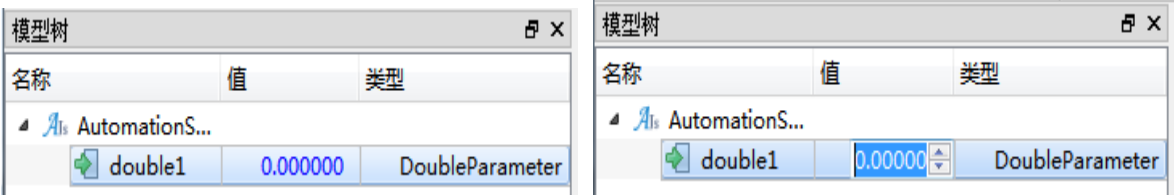


用户输入数值，即可更改。

1.4.3.3 双精度

双精度类型只接受浮点类型的数值，用户不能输入非数字的内容，比如“。abc\d”等字符。

修改数值的方法如下，在模型树上，选择对应的参数，点击“值”栏，鼠标点入后，界面变为如下图所示。

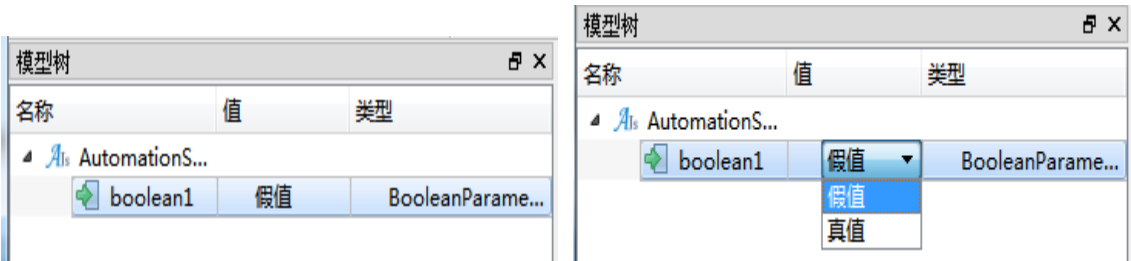


用户输入数值，即可更改。

1. 4. 3. 4布尔

布尔类型只接受真值和假值两种。

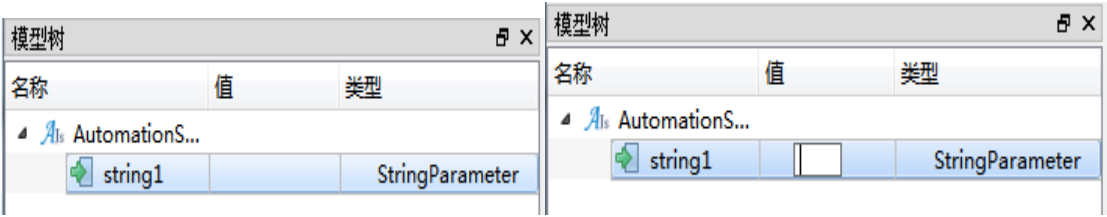
修改数值的方法如下，在模型树上，选择对应的参数，点击“值”栏，从下拉框内选择相应选项即可。界面如下图所示。



1. 4. 3. 5字符串

字符串类型是一串字符，接受中英文字符。

修改数值的方法如下，在模型树上，选择对应的参数，点击“值”栏，鼠标点入后，界面变为如下图所示。

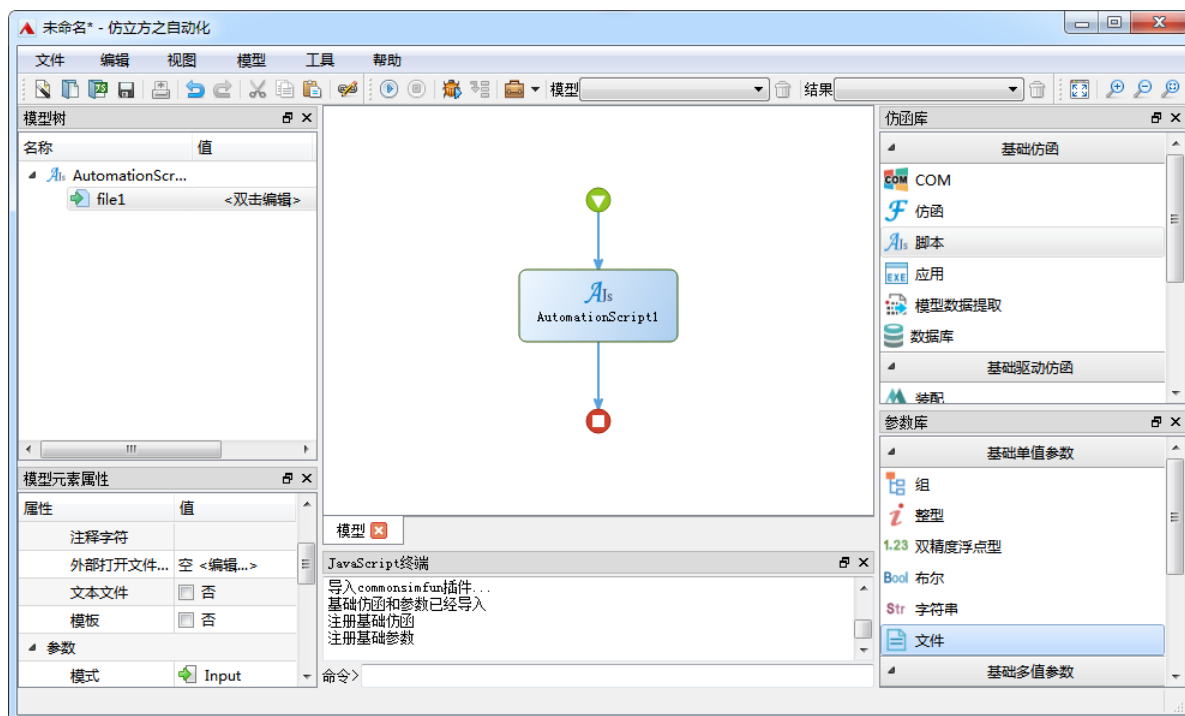


用户输入字符或文字，即可更改。

1. 4. 3. 6文件

文件参数只接受文本文件，并且可以提取文本文件中的数值参数。

文件参数的设置项如下图所示。



文件名称：点击浏览文本文件，设置对应的文件路径；文件路径设置完成后，可以双击模型树上节点，查看文件的内容；

注释符号：设置文本文件中注释行标识符；

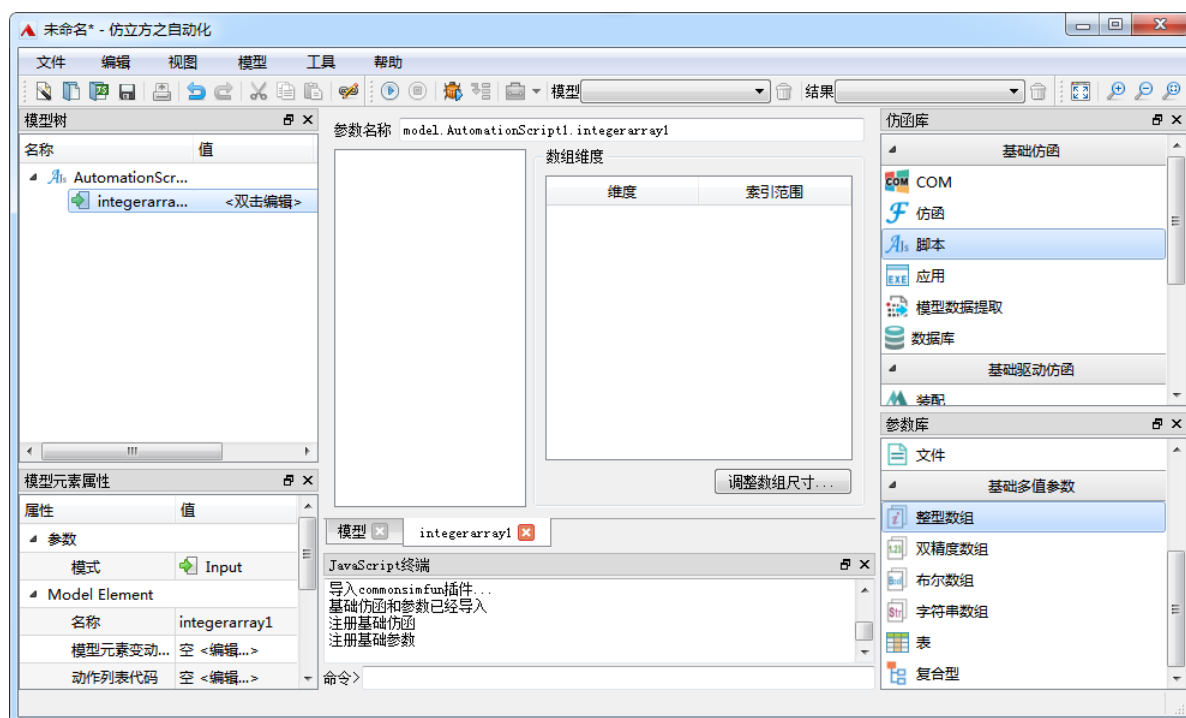
文件文件：设置文件是否为文件文件；

模板：设置此文件是否作为模板文件，模板文件可以提取参数。

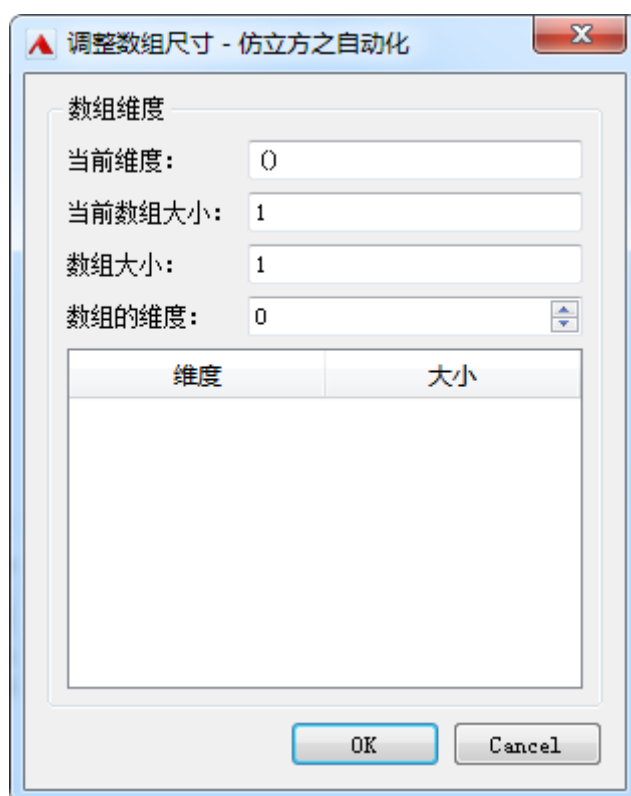
1.4.3.7 整数数组

整数数组类型只接受整数类型的数值，用户不能输入非数字的内容，比如“。abc\d”等字符。

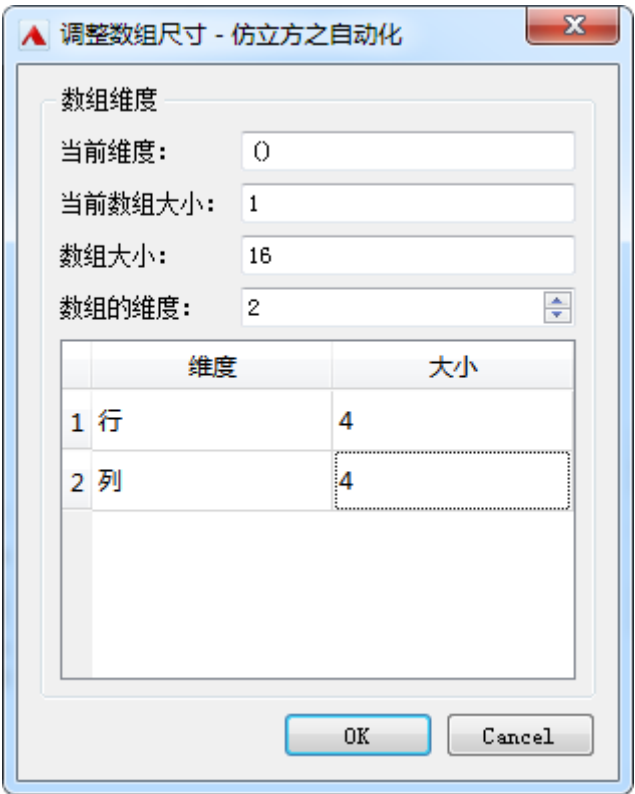
修改数组的方法如下，在模型树上，选择对应的参数，双击“值”栏，界面变为如下图所示。



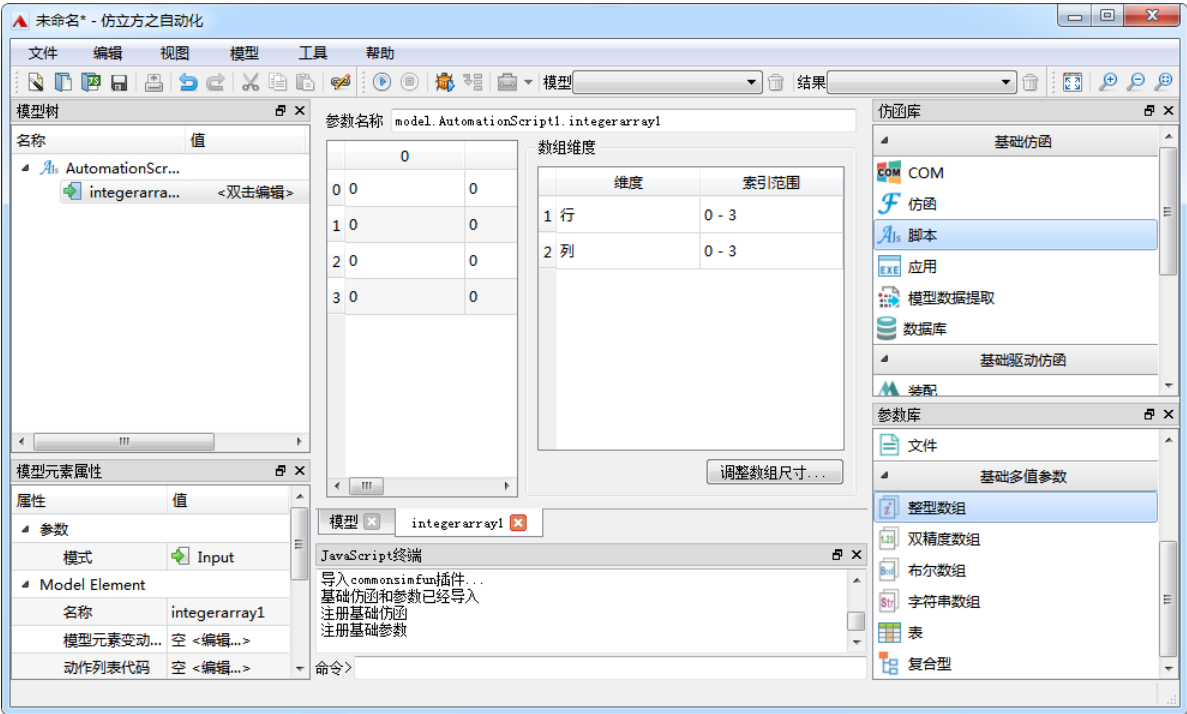
在右下角点击“调整数组尺寸”即可设置数组的维度，维度设置界面如下。



如果我们要生成一个 4×4 维的二维数组，在“数组维度”中输入 2，在 Dimension 中即可显示行、列的定义，将行、列的长度定义为 4，如下图所示。



点“OK”后，即完成数组的定义，如下图所示，用户可以进一步修改数组的数值。



1.4.3.8 双精度数组

双精度数组类型只接受浮点类型的数值，用户不能输入非数字的内容，比如“。abc\d”等字符。

修改数组的方法如下参见 1.4.3.7 一节。

1.4.3.9 布尔数组

布尔数组类型只接受 True 和 False 两种值。

修改数组的方法如下参见 1.4.3.7 一节。

1.4.3.10 字符串数组

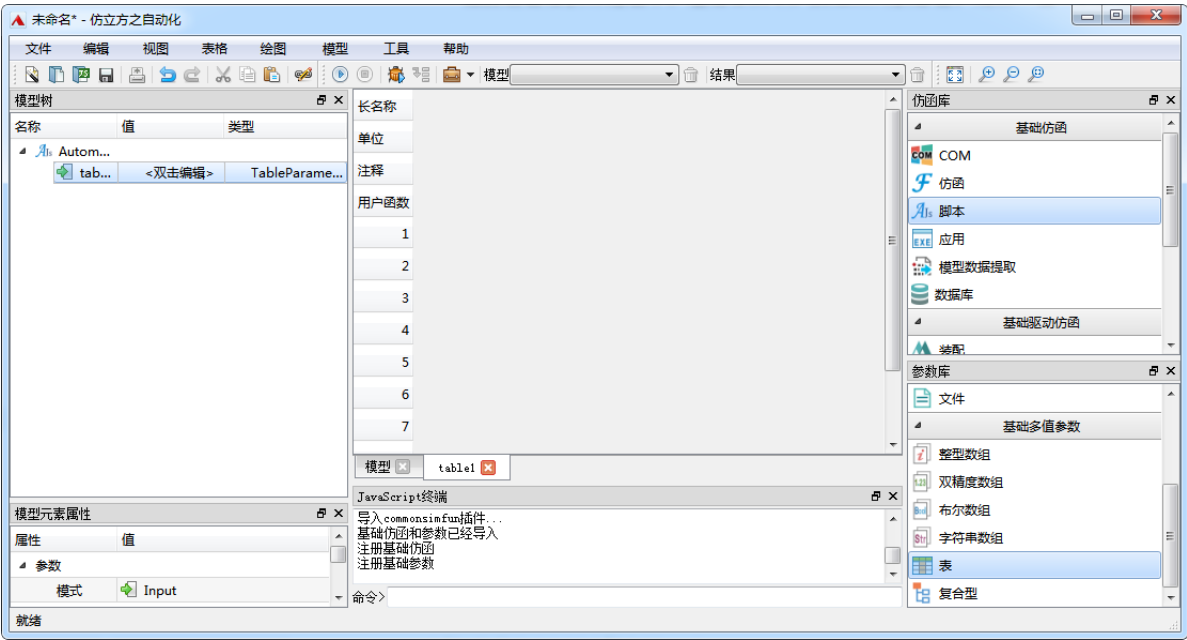
字符串数组类型是一组字符串，接受中英文字符。

修改数组的方法如下参见 1.4.3.7 一节。

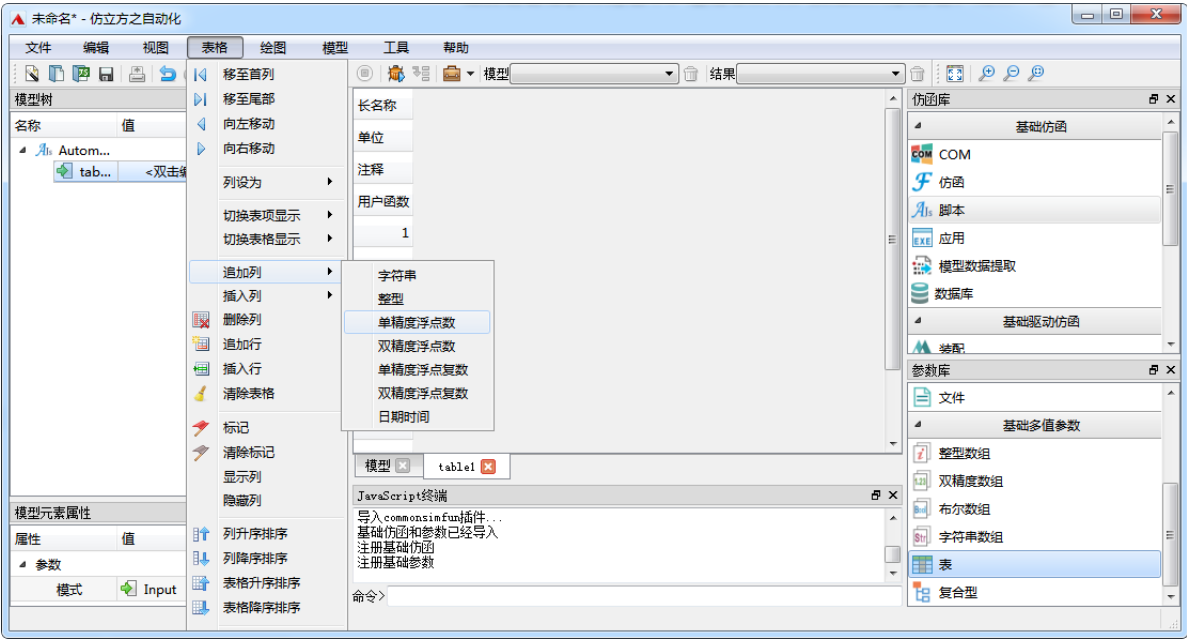
1.4.3.11 表

表是二维的，每列可以设置数据类型，每列可以支持若干行数据。

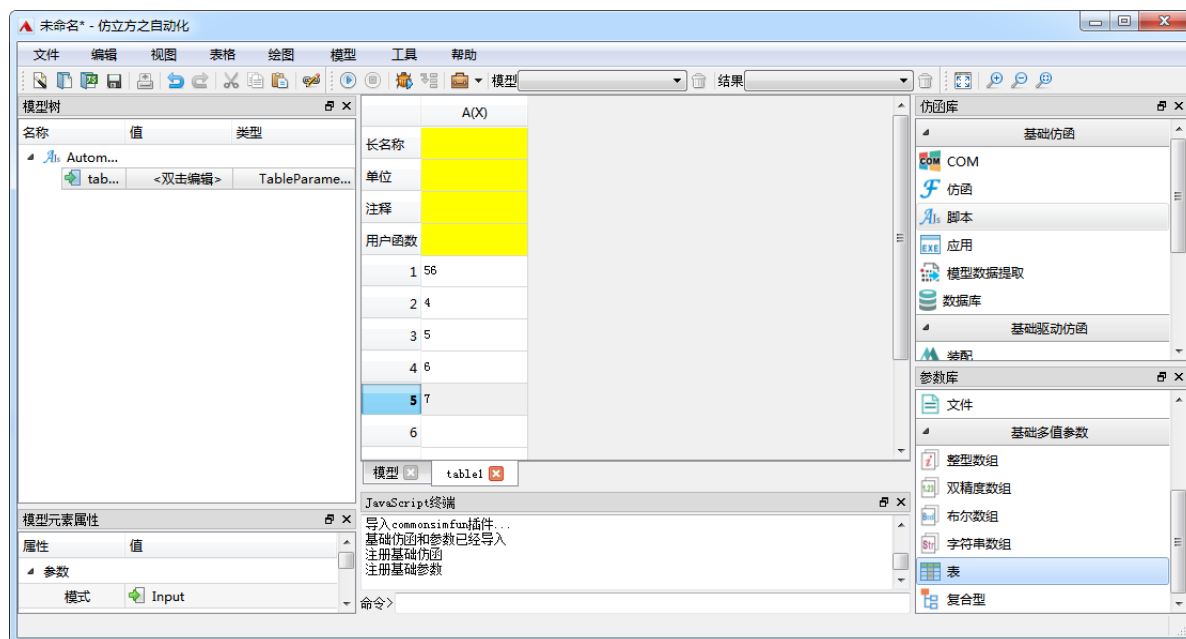
修改表数值的方法如下，在模型树上，选择对应的参数，双击“值”栏，界面变为如下图所示。



此时菜单栏多出了“表格”和“绘图”，点击“表格|追加列|单精度浮点数”，在表格中会增加一列，如下图所示。



用户点击单元格，输入相应数值，如下图所示。



关于“表格”和“绘图”菜单的功能说明，请参考《SimViz 用户手册》。

1.5 可视化链接参数

1.5.1 链接参数是什么


在仿真流程中，数据是重中之重，对数据需要统一的管理。

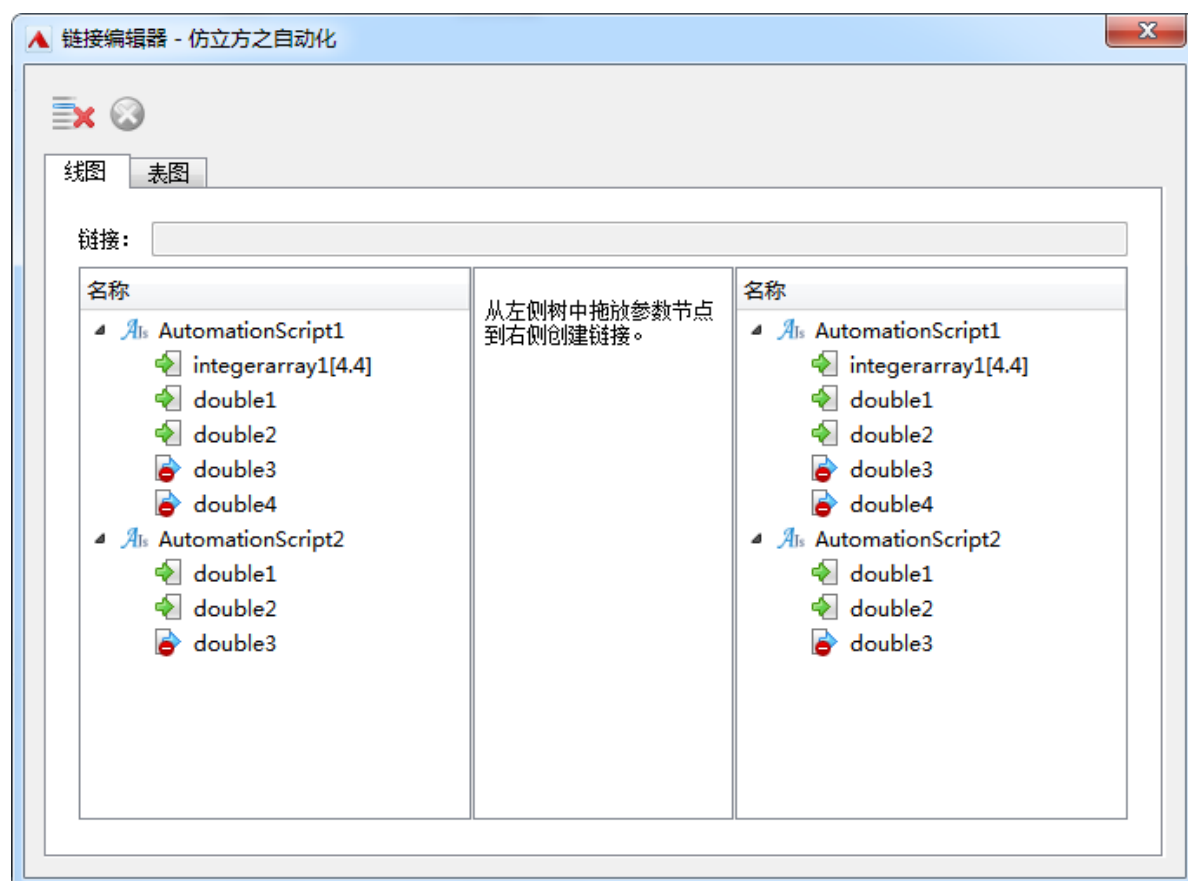
一个模型是由多个仿函组装的，多个仿函是组成一个模型的基础，但是如果没有仿函中参数之间的链接，那么模型中的各个仿函将会没有任何联系，是孤立的，形成不了一个有效的模型。

仿函中的参数链接表达了各个仿函之间的数据关系，一个仿函的输出参数是另一个或多个仿函的输入参数，在流程中通过参数链接把数据传递到不同的仿函之中，也即仿函底层的不同计算程序之中。

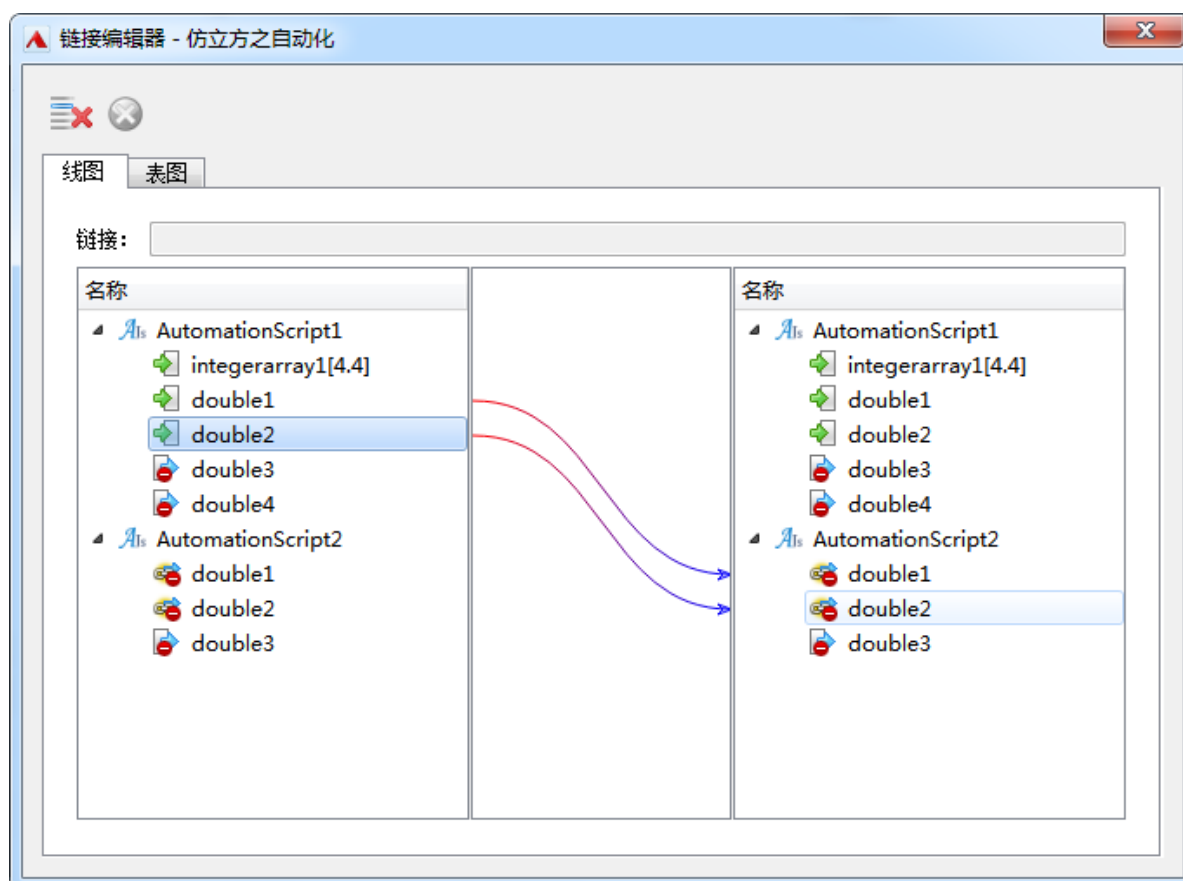
1.5.2 链接参数的操作

1.5.2.1 建立链接参数


在工具栏中点击 “链接编辑器”，弹出链接编辑器设置界面，如下图所示。



在“链接编辑器”中左右侧为树形窗口的树结构，自左向右拖拽参数建立链接，如下图所示。



1.5.2.2 删除链接参数

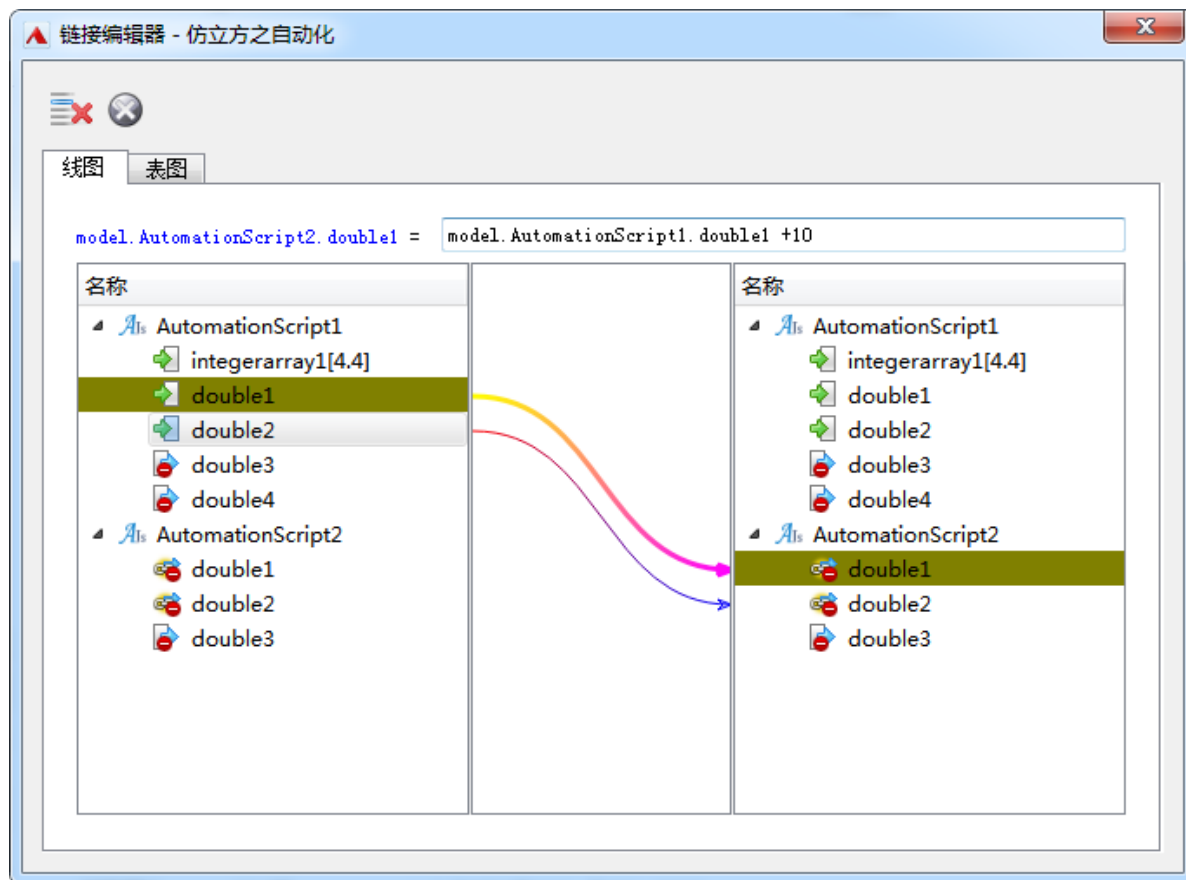
如果要删除某个链接，先选择待删除的链接，然后单击工具栏上的“删除”按钮，即可删除。

1.5.2.3 修改链接表达式

链接的功能不仅仅是传递一个数据，实际上它还能做的更多，因为链接可以是一个表达式，在这个表达式中可以把多个仿函数的多个参数综合起来，表达出一种复杂的计算关系。这也是由现实中的业务关系决定的，当业务关系复杂时，不能通过一个简单的数据传递表达各个不同模块的关系，而是要通过联系模块中的多个数据项组合成一个表达式才能达到业务的需求。

首先选择某个链接，此时“线图”中的文本输入框处于可编辑状态，用户可以更

改表达式，从而定义两个参数之间的数学关系。



2. 脚本建立模型

2.1 JavaScript 命名规则

采用 JavaScript 编写脚本，需要定义各种不同的变量，对于变量命名需要遵守一定的规则，具体规则如下：

- 1) 变量名是严格区分大小写的，如变量 abc 和 ABC 是两个变量，可以在程序分别对其进行声明，赋值和引用。
- 2) JavaScript 中变量名必须以字母、下划线 “_”、“\$” “中文” 打头，其余可以包括数字，字符，如 temp, _abc, example2 就是合法的变量名。
- 3) 不能引用 JavaScript 中的关键字作为变量，JavaScript 中定义了 40 多个关

键字，这些关键字都是 JavaScript 内部使用的，不能作为变量的名称，如 var, true, int, model 等不能作为变量名使用。

2.2 脚本建立驱动仿函

2.2.1 装配仿函

Assembly 可以理解为根仿函，在界面新建模型时，界面会默认创建一个 Assembly 仿函，但是脚本创建模型时必须指定根仿函，示例如下：

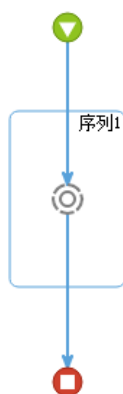
```
rom.model = new rom.Assembly('model', null);  
var model = rom.model;
```

2.2.2 序列仿函

这是一个复合仿函，可以包含其它子仿函，在模型执行的时候，控制其所有子仿函按顺序执行。示例如下：

```
rom.model = new rom.Assembly('model', null);  
var model = rom.model;  
  
var seq = new rom.Sequence('序列1', model);
```

在 Automation 中打开后的模型如下图所示。



2.2.3 并行仿函

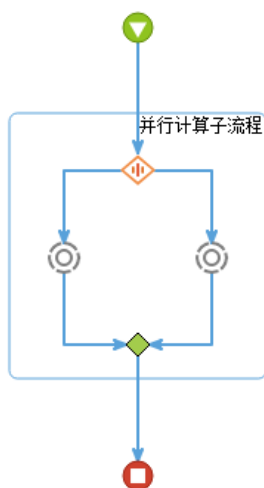
“并行”表示按并行依次执行内部的仿函，在“并行”仿函中可以添加新的驱动

仿函和模型仿函。新建“并行”仿函默认只建立两个分支，当两个分支上都有仿函时，可以将仿函添加到第三个分支上。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
// 并行仿函

var parallel = new rom.Parallel('并行计算子流程', model);
```

在Automation中打开后的模型如下图所示。



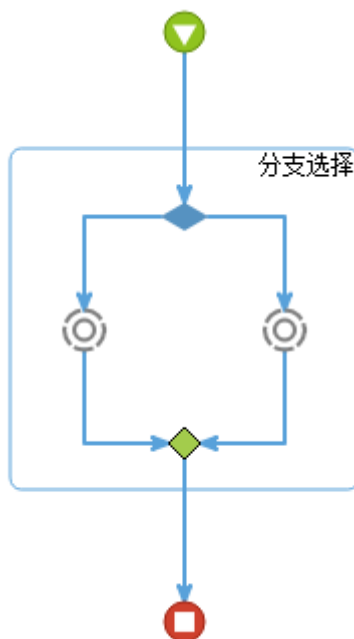
2.2.4 分支仿函

“分支”表示按照某种条件执行某个分支上的仿函，在“分支”仿函中可以添加新的驱动仿函和模型仿函。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var sw = new rom.Switch('分支选择', model);
var data1 = new rom.IntegerParameter('data1', sw);

sw.setBranchCondition(srArr[0], data1 <= 25);
```

在Automation中打开后的模型如下图所示。



2.2.5 总线仿函

“总线”表示按某些条件来循环执行内部的仿函，当某个仿函条件满足，就执行；否则不执行。在“总线”仿函中可以添加新的驱动仿函和模型仿函。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;

var r0 = new rom.Bus('总线控制', model);
var r1 = new rom.AutomationScript('脚本一', r0);
var p1 = new rom.DoubleParameter('data1', rom.Input, r1);
var p2 = new rom.DoubleParameter('data2', rom.Output, r1);
r1.scriptCode = "data2 = Math.sin(data1) + Math.random();";

var r2 = new rom.AutomationScript('脚本二', r0);
var p3 = new rom.DoubleParameter('data1', rom.Input, r2);
var p4 = new rom.DoubleParameter('data2', rom.Output, r2);
r2.scriptCode = "data2 = Math.cos(data1) + Math.random();";

var r3 = new rom.AutomationScript('脚本三', r0);
var p5 = new rom.DoubleParameter('data1', rom.Input, r3);
var p6 = new rom.DoubleParameter('data2', rom.Output, r3);
```



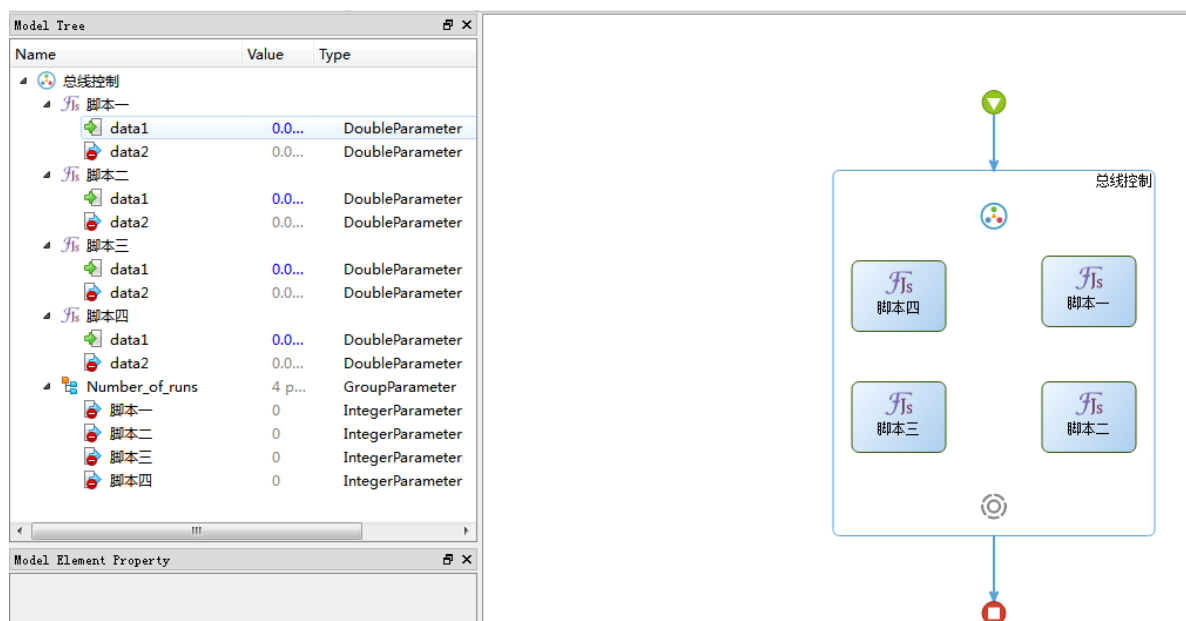
```

r3.scriptCode = "data2 = Math.sin(data1) + Math.random();";

var r4 = new rom.AutomationScript('脚本四', r0);
var p7 = new rom.DoubleParameter('data1', rom.Input, r4);
var p8 = new rom.DoubleParameter('data2', rom.Output, r4);
r4.scriptCode = "data2 = Math.cos(data1) + Math.random();";

```

在Automation中打开后的模型如下图所示。



2.2.6 循环仿函

“循环”表示循环执行内部的仿函若干次，在“循环”仿函中可以添加新的驱动仿函和模型仿函。“循环”仿函内部会有一个变量值counter，表示循环执行的次数。

示例如下：

```

rom.model = new rom.Assembly('model', null);
var model = rom.model;
var loop = new rom.Loop('条件判断子流程', model);
// 循环类型
loop.loopType = rom.Loop.For;
// 循环参数
loop.setLoopParameterByGeneralParameter(input1);

```

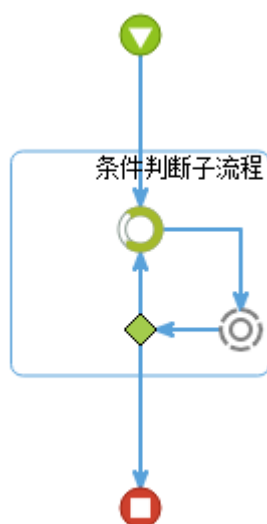
```

loop.setFrom(0);
loop.setTo(100);

loop.setIncrement(1);

```

在Automation中打开后的模型如下图所示。



2.3 脚本建立基础仿函

2.3.1 COM 仿函

“COM”仿函可以直接调用COM组件中的函数，供基础仿函或流程仿函使用。在代码中建立COM仿函的示例如下。

```

rom.model = new rom.Assembly('model', null);
var model = rom.model;

var com = new rom.COM('COM 组件', model);
var double1 = new rom.DoubleParameter('double1', rom.Input, com);
double1.value = 12.12;
var double2 = new rom.DoubleParameter('double2', rom.Input, com);
double2.value = 23.12;
var double3 = new rom.DoubleParameter('double3', rom.Output, com);

com.interactFunction = function() {

```

```

self.loadCOMObject(' {9F6F46E3-2F6B-4EA7-A68F-E1E697E2FC96} ');
//self.loadCOMObject(' {16D0F7D6-F641-4362-A549-FC6436556794} ');
process.print(' com is null: ', self.isNull());
//获取 com 接口中的属性
//process.print(self.comProperty('PI'));
//调用 com 接口中的方法，第一个参数是 com 接口方法的原型，第二个参数
是一个数组，
//将传递给 com 接口方法的参数.
var res = self.callComFunction(' AddNum(double, double)',
    [self.parameters.double1.value,
self.parameters.double2.value]); // 传入输入参数
process.print(typeof res);
process.print(res);
self.parameters.double3.value = res; // 设置输出参数
};

```

在Automation中打开后的模型如下图所示。



2.3.2 仿函

“仿函”没有具体的功能，只是起到占位符的作用。在SimCube.Automation中使用Javascript脚本建立模型示例如下：

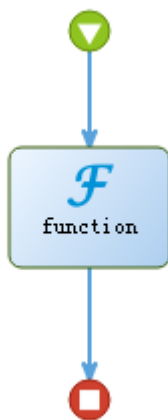
```

rom.model = new rom.Assembly('model', null);
var model = rom.model;

```

```
var myfun = new rom.SimFunction('function', model);
```

在 Automation 中打开后的模型如下图所示。



2.3.3 脚本仿函

“脚本”可以完成定制编写的Javascript代码。Javascript是通用的脚本语言，功能强大，使用简便，用户可以方便地在SimCube.Automation中使用Javascript脚本来定制某项功能。示例如下：

```
rom.model = new rom.Assembly('model', null);  
var model = rom.model;  
  
var as = new rom.AutomationScript('as', model);
```

在Automation中打开后的模型如下图所示。



2.3.4 应用仿函

“应用”仿函主要用来封装用户编写的HomeIn程序，并调用执行，也可以调用商业软件。示例如下：

```
rom.model = new rom.Assembly('model', null);  
var model = rom.model;  
var sa = new rom.SimApp('sa', model);  
sa.appPath = 'E:/code/ aeroarg.exe';
```

在Automation中打开后的模型如下图所示。



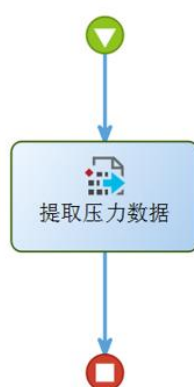
2.3.5 模型数据提取仿函

“模型数据提取”仿函可以从 rom 文件中提取出数据，供基础仿函或流程仿函使

用。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var dataextract = new rom.ModelDataExtract('提取压力数据', model);
dataextract.parameters.inputs.parameterFromName('scriptFile').setFileName('%TEMPLATE%/ romdataextract.js');
```

在Automation中打开后的模型如下图所示。



2.3.6 数据库仿函

“数据库”仿函可以从数据库文件中提取出数据，供基础仿函或流程仿函使用。

示例如下：

```
/*
 * 数据库仿函示例
 * 使用 sqlite 数据库，在当前脚本所在路径创建一个文件名是 abc.db3 的数据库。
 * 数据库中只包含一个表，存储人名，年龄与体重信息。
 */
rom.model = new rom.Assembly('model', null);
var model = rom.model;

// 若没有数据库，创建一个 sqlite 数据库，返回数据库中的姓名
var allnames = function() {
    var persons = [['张三', 21, 60],
                   ['赵四', 22, 61],
```

```
        ['王五', 32, 81],
        ['马六', 12, 41]];

var names = [];

for (var i = 0; i < persons.length; ++i)
    names.push(persons[i][0]);

var dbfile = path.join(path.dirname(__FILE__), 'abc.db3');
if (!fs.exists(dbfile)) {
    process.print('create db');
    var db = sql.addDatabase('SQLITE', 'persons');
    db.databaseName = dbfile;
    if (!db.open()) {
        process.print('open db failed. ');
        sql.removeDatabase('persons');
        return;
    }
    var table = 'CREATE TABLE persons (' +
        'id INTEGER PRIMARY KEY AUTOINCREMENT,' +
        'name VARCHAR(40) NOT NULL,' +
        'age INTEGER NOT NULL,' +
        'weight INTEGER NOT NULL);';
    var q = db.exec(table);
    process.print(q.executedQuery);
    q.prepare('INSERT INTO persons (name, age, weight)' +
        'VALUES (:name, :age, :weight)');
    for (var i = 0, size = persons.length; i < size; ++i) {
        q.bindValue(':name', persons[i][0]);
        q.bindValue(':age', persons[i][1]);
        q.bindValue(':weight', persons[i][2]);
        q.exec();
    }
    process.print(q.executedQuery);
    db.close();
    sql.removeDatabase('persons');
}
```

```
        return names;
    }();

var dbSf = new rom.Database('数据库', model);
dbSf.driverName = 'SQLite'; // 设置数据库驱动器
dbSf.connectionName = dbSf.name; // 设置数据库连接名称
dbSf.databaseName = path.join(path.dirname(__FILE__), 'abc.db3'); // 设置数据库名称, 对于 sqlite 数据库即文件名称

var name = new rom.StringParameter('Name', rom.Input, dbSf);
name.enumEnabled = true;
name.description = '姓名';
name.enumValues = allnames;
name.enumIndex = 0;

var age = new rom.IntegerParameter('Age', rom.Output, dbSf);
age.units = '岁';
age.description = '年龄';
var weight = new rom.IntegerParameter('Weight', rom.Output, dbSf);
weight.units = '公斤';
weight.description = '体重';

dbSf.interactFunction = function() { // 数据库仿函数执行时执行的 JavaScript 代码
    var db = self.openedDatabase; // 如果上面的设置正确, 此时数据库已经打开
    var stat = 'SELECT age, weight FROM persons ' +
        'WHERE name = :name;';
    var q = db.exec(); // 从数据库对象中获得一个查询对象
    q.prepare(stat);
    q.bindValue(':name', self.parameters.Name.value);
    q.exec(); // 执行查询
    if (q.first()) {
        process.print(q.value(0), q.value('weight'));
        self.parameters.Age.value = q.value(0); // 从查询结果中设置输出参数值
        self.parameters.Weight.value = q.value(1);
    }
}
```



```
};
```

在Automation中打开后的模型如下图所示。



2.4 脚本建立参数仿函

2.4.1 组

组可以将不同类型的参数打包成集合，组下面不能包含组，但可以包含其它的各种类型的参数。参数应添加到基础仿函中，示例如下：

```
rom.model = new rom.Assembly('model', null);  
var model = rom.model;  
var script = new rom.AutomationScript('script', model);  
var com1 = new rom.CompositeParameter('com1', rom.Input, script);  
var data1 = new rom.IntegerParameter('data1', rom.Input, com1);  
var data2 = new rom.DoubleParameter('data2', rom.Input, com1);  
var data3 = new rom.BooleanParameter('data3', rom.Input, com1);  
var data4 = new rom.StringParameter('data4', rom.Input, com1);  
  
var file1 = new rom.FileParameter('file1', rom.InputFile, com1);
```

在Automation中打开后的模型如下图所示。

模型树			
名称	值	描述	类型
script			
com1	5子参数		CompositeParameter
data1	0		IntegerParameter
data2	0.000000		DoubleParameter
data3	假值		BooleanParameter
data4			StringParameter
file1	<双击编...		FileParameter

2.4.2 整数

整数类型只接受整数类型的数值，用户不能输入非数字的内容，比如“。abc\d”等字符。参数应添加到基础仿函中，示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var script = new rom.AutomationScript('script', model);
var data1 = new rom.IntegerParameter('data1', rom.Input, script);
```

在Automation中打开后的模型如下图所示。

模型树			
名称	值	描述	类型
script			
data1	0		IntegerParameter

2.4.3 双精度

双精度类型只接受浮点类型的数值，用户不能输入非数字的内容，比如“。abc\d”等字符。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var script = new rom.AutomationScript('script', model);
var data1 = new rom.DoubleParameter ('data1', rom.Input, script);
```

在Automation中打开后的模型如下图所示。

模型树			
名称	值	描述	类型
script			
data1	0.000000		DoubleParameter

2.4.4 布尔

布尔类型只接受True和False两种值。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var script = new rom.AutomationScript('script', model);
var data1 = new rom.BooleanParameter('data1', rom.Input, script);
```

在Automation中打开后的模型如下图所示。

Model Tree			
Name	Value	Type	Des
script			
boolean1	FALSE	BooleanParameter	

2.4.5 字符串

字符串类型是一串字符，接受中英文字符。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var script = new rom.AutomationScript('script', model);
var data1 = new rom.StringParameter('data1', rom.Input, script);
```

在Automation中打开后的模型如下图所示。

模型树			
名称	值	描述	类型
script			
data1			StringParameter

2.4.6 文件

字符串类型是一串字符，接受中英文字符。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var script = new rom.AutomationScript('script', model);
var file1 = new rom.FileParameter('file1', rom.InputFile, script);
```

在Automation中打开后的模型如下图所示。

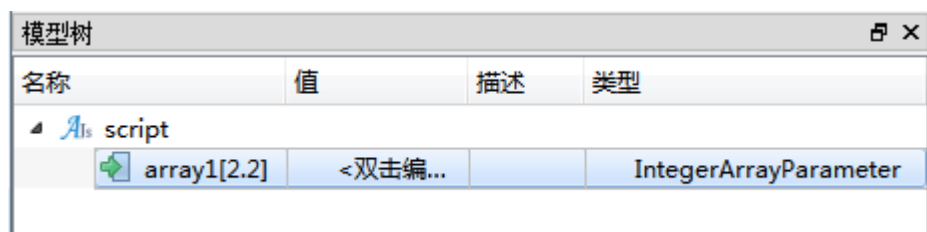
模型树			
名称	值	描述	类型
script			
file1	<双击编...		FileParameter

2.4.7 整数数组

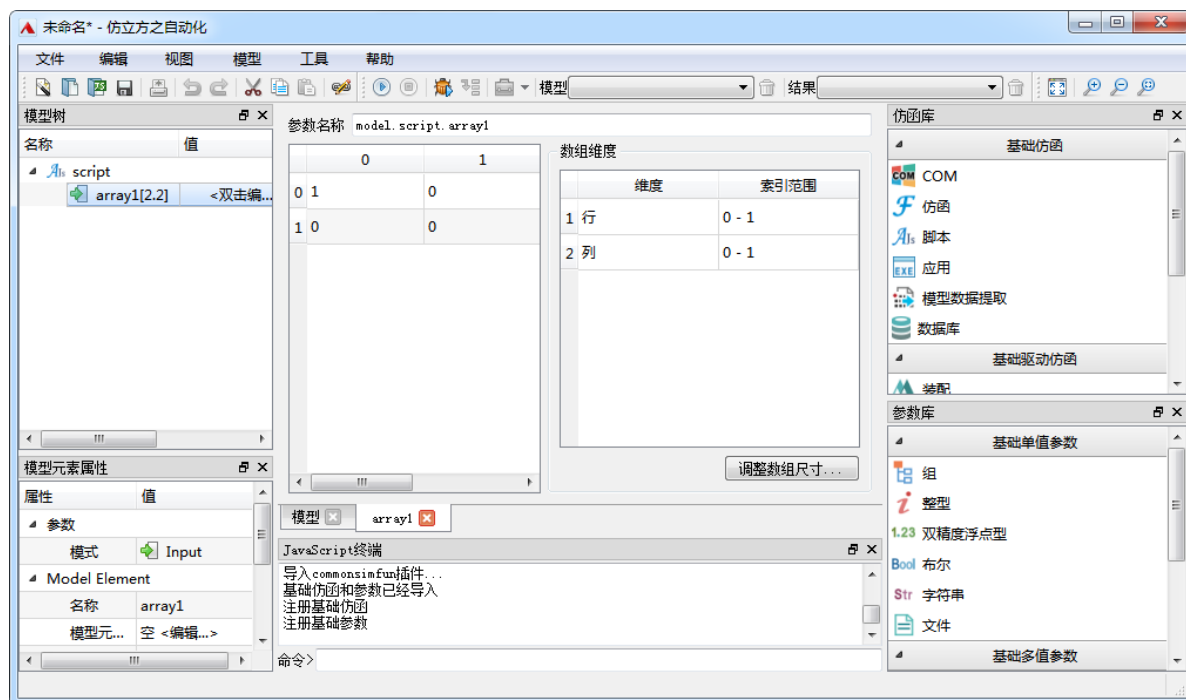
整数数组类型只接受整数类型的数值，用户不能输入非数字的内容，比如“..abc\d”等字符。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var script = new rom.AutomationScript('script', model);
var array1 = new rom.IntegerArrayParameter('array1', rom.Input, [2, 2], script);
array1.setValue([0, 0], 1);
```

在Automation中打开后的模型如下图所示。



当双击 array1 的值列，可以看到设置的数组值，如下图所示。



2.4.8 双精度数组

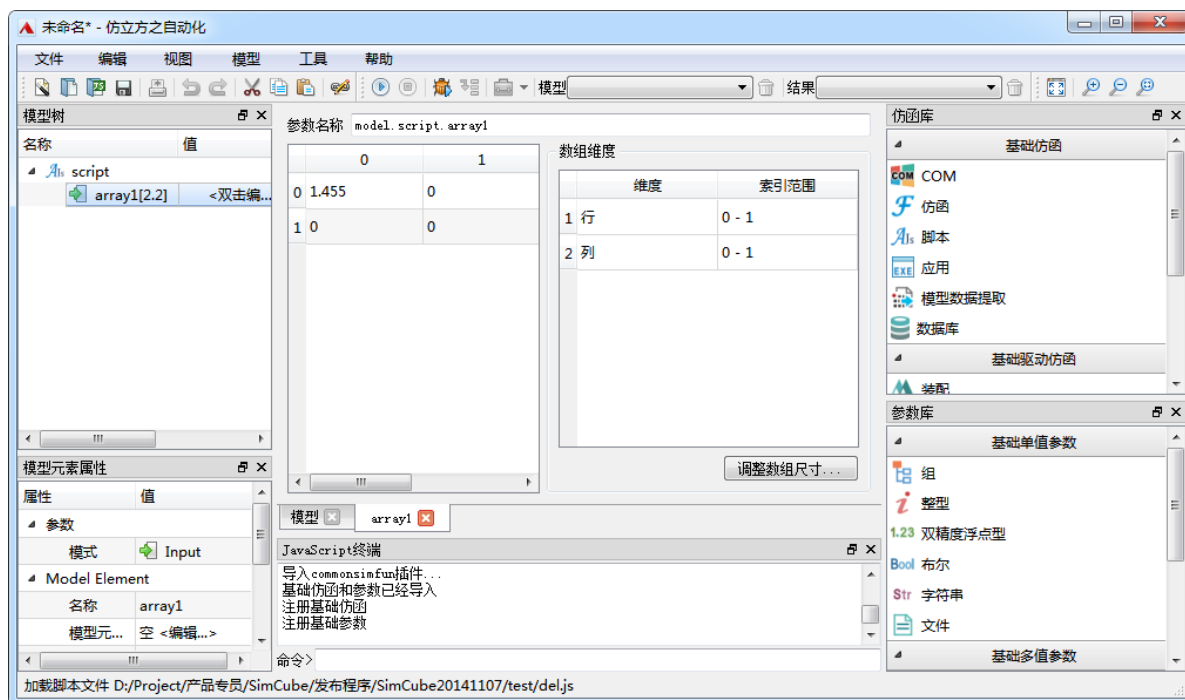
双精度数组类型只接受浮点类型的数值，用户不能输入非数字的内容，比如“。abc\d”等字符。示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var script = new rom.AutomationScript('script', model);
var array1 = new rom.DoubleArrayParameter('array1', rom.Input, [2, 2], script);
array1.setValue([0, 0], 1.455);
```

在Automation中打开后的模型如下图所示。

模型树			
名称	值	描述	类型
script			
array1[2.2]	<双击编...		DoubleArrayParame...

当双击 array1 的值列，可以看到设置的数组值，如下图所示。



2.4.9 布尔数组

布尔数组类型只接受 True 和 False 两种值。

示例如下：

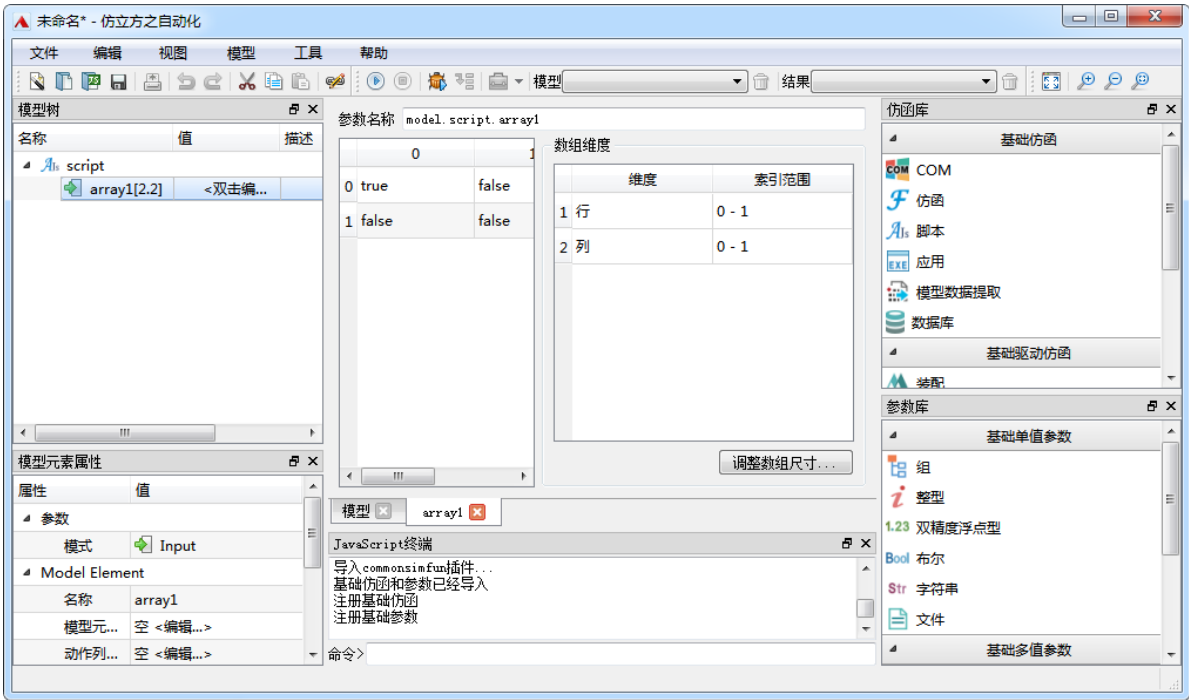
```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var script = new rom.AutomationScript('script', model);
var array1 = new rom.BooleanArrayParameter('array1', rom.Input, [2, 2], script);

array1.setValue([0, 0], "TRUE");
```

在Automation中打开后的模型如下图所示。

模型树			
名称	值	描述	类型
script			
array1[2.2]	<双击编...		BooleanArrayParam...

当双击 array1 的值列，可以看到设置的数组值，如下图所示。



2.4.10 字符串数组

字符串数组类型是一组字符串，接受中英文字符。

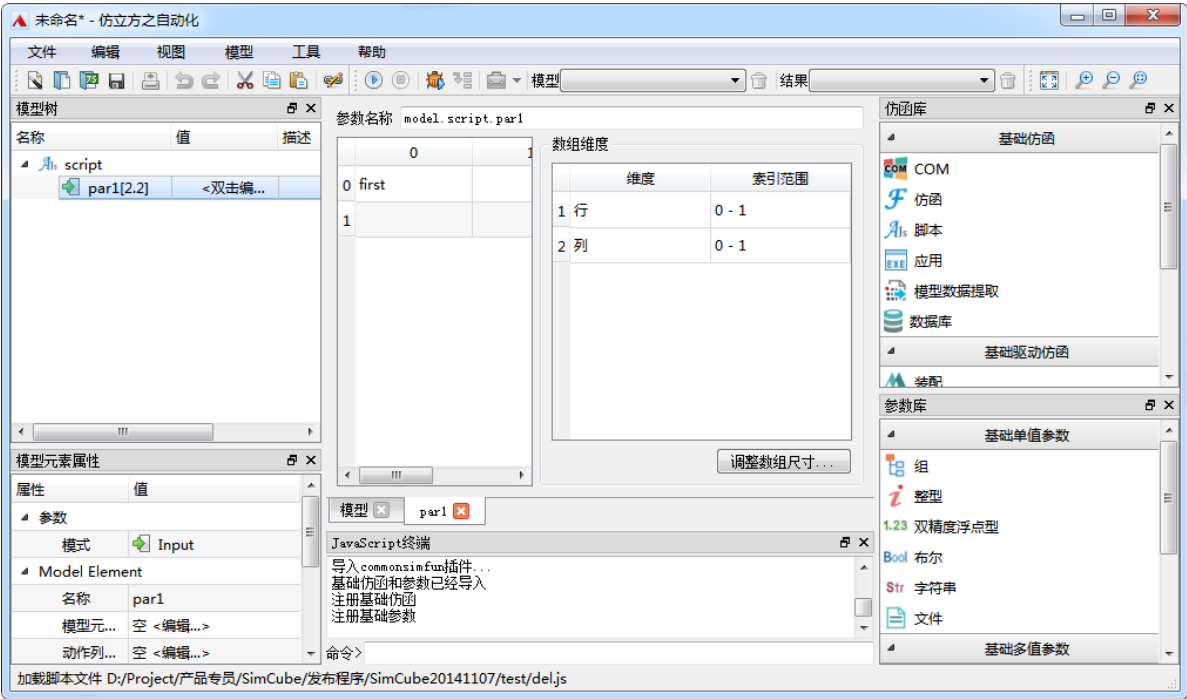
示例如下：

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;
var script = new rom.AutomationScript('script', model);
var array1 = new rom.IntegerArrayParameter('array1', rom.Input, [2,2], script);
array1.setValue([0,0], 1);
```

在Automation中打开后的模型如下图所示。

模型树			
名称	值	描述	类型
script	par1[2.2]	<双击编...	StringArrayParameter

当双击 par1 的值列，可以看到设置的数组值，如下图所示。



2.4.11 表

表是二维的，每列可以设置数据类型，每列可以支持若干行数据。示例如下。

```
rom.model = new rom.Assembly('model', null);
var model = rom.model;

var as = new rom.AutomationScript('as', model);

var data1 = new rom.TableParameter('data1', rom.Input, as);
data1.appendColumn(rom.TableParameter.Integer, 3);
data1.appendColumn(rom.TableParameter.Float, 3);
data1.appendColumn(rom.TableParameter.Double, 3);
data1.appendColumn(rom.TableParameter.String, 3);
data1.appendColumn(rom.TableParameter.DateTime, 3);
```



```
data1.appendColumn(rom.TableParameter.FloatComplex, 3);
data1.appendColumn(rom.TableParameter.DoubleComplex, 3);

data1.fillColumnAt(0, [1, 2, 3]);
data1.setColumnNameAt(0, 'col1');
data1.setColumnLongNameAt(0, 'column 1');
data1.setColumnUnitAt(0, '秒');
data1.setColumnCommentAt(0, '整数列');
data1.setColumnSignificanceAt(0, rom.TableParameter.X);
data1.appendRowAt(0, 7928);
data1.insertRowAt(0, 1, 7128);

data1.fillColumnAt(1, [4.12, .125, -12.6]);
data1.setColumnNameAt(1, 'col2');
data1.setColumnLongNameAt(1, 'column 2');
data1.setColumnUnitAt(1, '秒');
data1.setColumnCommentAt(1, '浮点数列');
data1.setColumnSignificanceAt(1, rom.TableParameter.Y);
data1.appendRowAt(1, 79.28);
data1.insertRowAt(1, 1, 71.28);

data1.fillColumnAt(2, [4.12, .125, -12.6]);
data1.setColumnCommentAt(2, '双精度浮点数列');
data1.appendRowAt(2, 79.28);
data1.insertRowAt(2, 1, 71.28);

data1.fillColumnAt(3, ['abc', 'Hello World', 'Bye']);
data1.setColumnCommentAt(3, '字符串列');
data1.appendRowAt(3, "79.28");
data1.insertRowAt(3, 1, "71.28");

data1.fillColumnAt(4, [new Date(2014, 1, 2, 12, 23),
                      new Date(2011, 11, 3), new Date(2012, 10, 12)]);
data1.setColumnCommentAt(4, '日期时间列');
data1.appendRowAt(4, new Date(2014, 11, 2));
```

```

data1.insertRowAt(4, 1, new Date(2014, 11, 3));

data1.fillColumnAt(5, [[1.1, 2.2], [3.3, 4.4], [5.5, 6.6]]);
data1.setColumnCommentAt(5, '浮点数复数列');
data1.appendRowAt(5, [79, 28]);
data1.insertRowAt(5, 1, [79, 18]);

data1.fillColumnAt(6, [[1.1, 2.2], [3.3, 4.4], [5.5, 6.6]]);
data1.setColumnCommentAt(6, '双精度浮点数复数列');
data1.appendRowAt(6, [79, 28]);
data1.insertRowAt(6, 4, [79, 18]);

var data2 = new rom.TableParameter('data2', rom.Output, as);

data1.swap(3, 5);
data1.swap(4, 6);

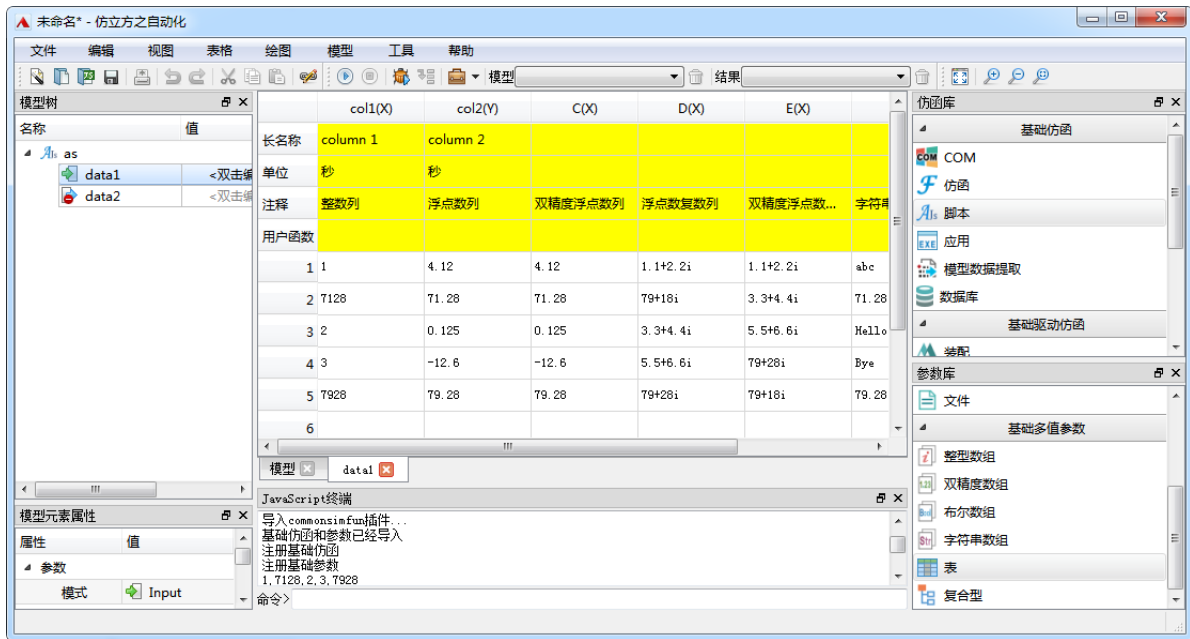
log(data1.integerColumnAt(0));
log(data1.floatColumnAt(1));
log(data1.doubleColumnAt(2));
log(data1.floatComplexColumnAt(3));
log(data1.doubleComplexColumnAt(4));
log(data1.stringColumnAt(5));
log(data1.dateTimeColumnAt(6));

```

在Automation中打开后的模型如下图所示。

模型树			✖
名称	值	类型	
as			
 data1	<双击编辑>	TableParameter	
 data2	<双击编辑>	TableParameter	

当双击 data1 的值列，可以看到设置的值，如下图所示。



2.5 脚本链接参数

仿函中的参数链接表达了各个仿函之间的数据关系，一个仿函的输出参数是另一个或多个仿函的输入参数，在流程中通过参数链接把数据传递到不同的仿函之中，也即仿函底层的不同计算程序之中，示例如下。

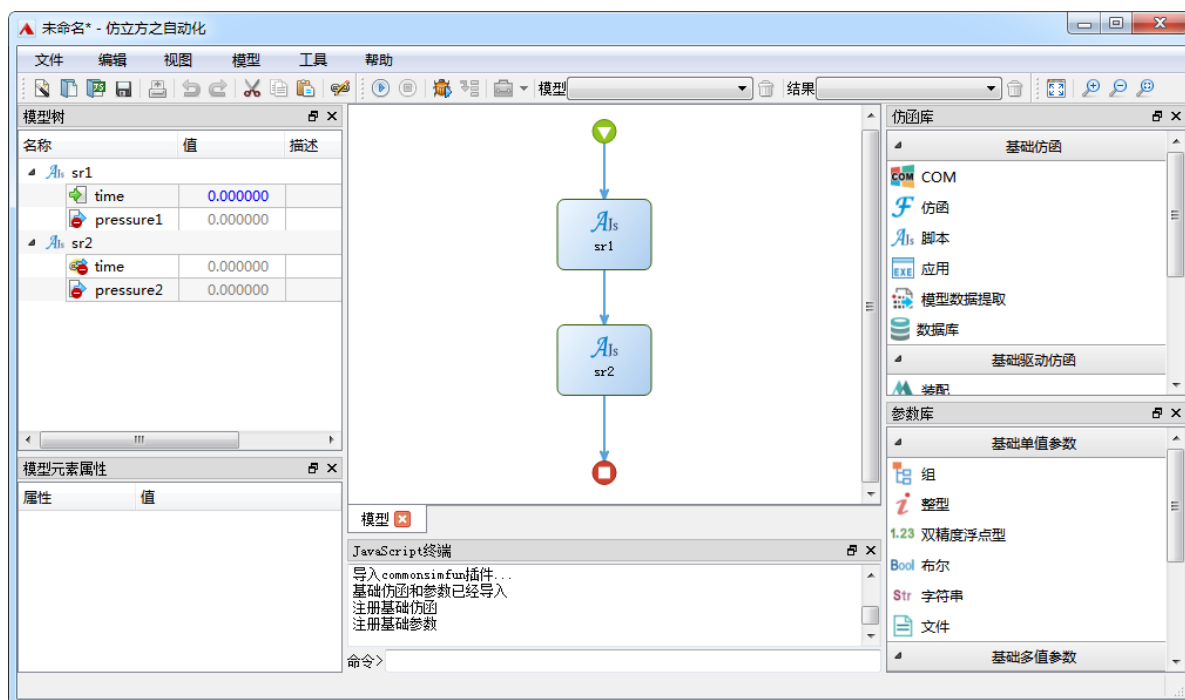
```
rom.model = new rom.Assembly('model', null);
var model = rom.model;


var r1 = new rom.AutomationScript('sr1', model);
var p1 = new rom.DoubleParameter('time', rom.Input, r1);
var p2 = new rom.DoubleParameter('pressure1', rom.Output, r1);

var r2 = new rom.AutomationScript('sr2', model);
var p3 = new rom.DoubleParameter('time', rom.Input, r2);
var p4 = new rom.DoubleParameter('pressure2', rom.Output, r2);

r2.createLinkByParameters(p3, p1);
```

在Automation中打开后的模型如下图所示。



在工具栏中点击 “链接编辑器”，弹出链接编辑器设置界面，如下图所示。

