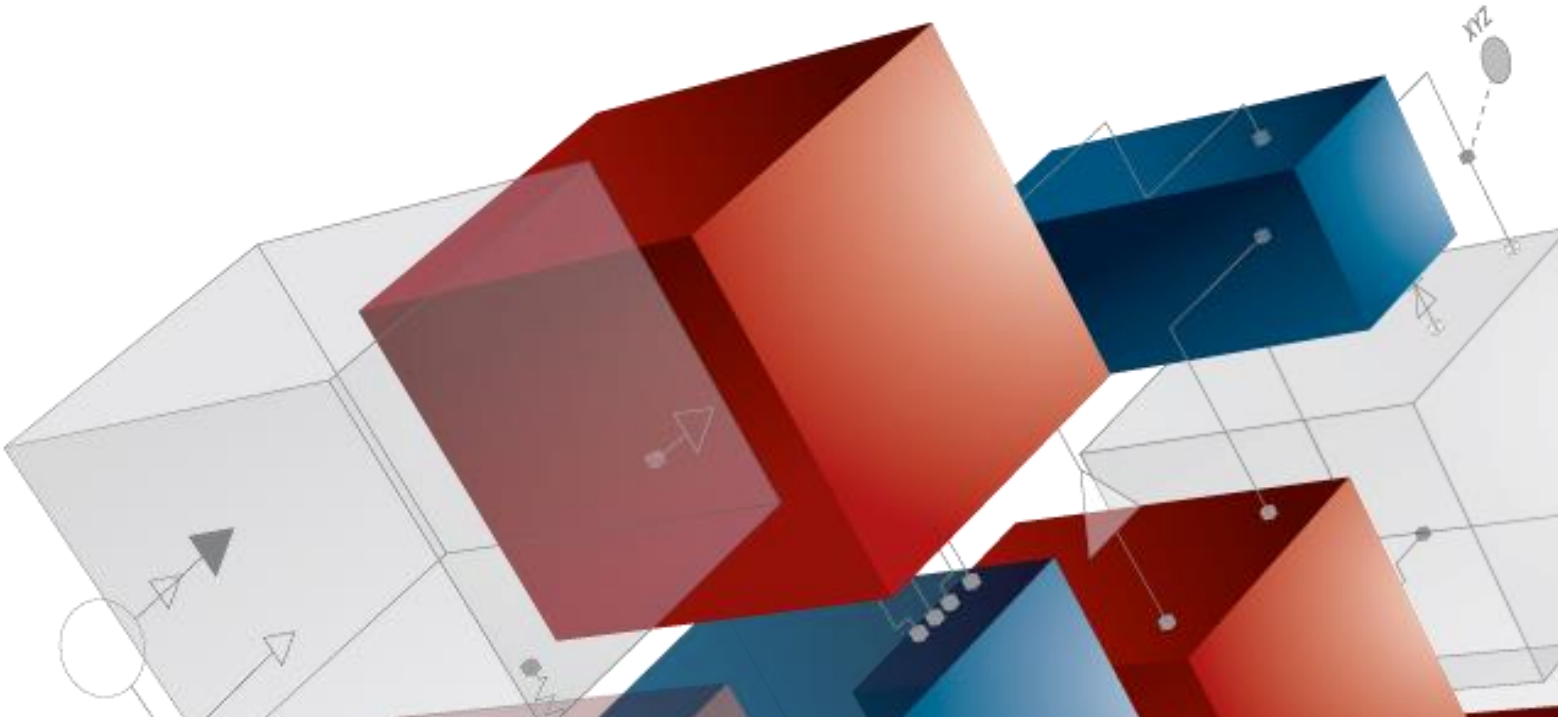
多领域系统建模仿真软件
MapleSim 实践教程

赵 跃　徐俊林 ◎ 主编

MAPLESOFT
www.maplesoft.com.cn



前言

本教程是 Maplesoft 公司为广大 MapleSim 软件用户编制的入门级使用教程，供大家免费使用和参考。

本教程使用的建模软件是 Maple 和 MapleSim 2016.0，用户需要提前安装以下软件：

- Maple 2016 及以上版本
- MapleSim 2016 及以上版本
- MapleSim CAD Toolbox2016 及以上版本
- MapleSim Driveline Library 2016 及以上版本
- MapleSim Control Design Toolbox 2016 及以上版本
- MapleSim Connector 2016 及以上版本

如需申请试用版软件，请登录 Maplesoft 网站（www.maplesoft.com.cn）申请。

由于水平有限，教程中错误之处在所难免，敬请读者指正！

2017 年 7 月 4 日

Maplesoft 中国

目录

介绍

■ MapleSim 概述

MapleSim 是一个创建和模拟复杂多领域物理系统的建模环境，使用代表物理系统的框图建模。求解器使用符号和数值混合求解器，MapleSim 可以自动生成模型框图对应的模型方程，并实现高保真仿真。

建立复杂多领域模型

建立 MapleSim 模型时，用户可以将来自不同工程领域的建模元件集成在一起组成一个完整的系统。MapleSim 预置一个包含数百个建模元件的库，例如电气、液压、机械、热力设备，传感器和源，信号模块等。用户也可以创建自定义元件满足建模和仿真的需要。

高级符号和数值计算功能

MapleSim 使用了数学软件 Maple 中的高级符号和数值功能，生成数学模型模拟物理系统的行为。因此，可以将符号化简技术应用到方程，生成简洁和数值高效的模型。

预置的分析工具和模板

Maple 通过 Maple 工作表的形式提供不同种类的预置分析工具和模板，例如参数优化。为了交互方式分析模型和呈现结果，用户可以使用 Maple 中的工具，例如嵌入式 GUI 控件、绘图工具、文件创建工具等。用户也可以转换模型为 C 代码，与其他应用程序和工具的协同工作，包括实时仿真。

交互式 3D 可视化工具

在 MapleSim 三维可视化环境中可以建立多体机构系统的模型，用于验证模型的三维配置、在不同的模型设置条件和仿真起始时间下直观地分析模型行为。

■ 相关工具

MapleSim 2019，需要 Maple 2019.

Maplesoft 提供一系列工具箱、附加模块、以及其他应用，拓展 Maple 和 MapleSim 实现工程设计项目的能力和范围。更多信息见：https://www.maplesoft.com.cn/products

■ 相关资源

资源	描述
MapleSim帮助系统	提供以下信息：

	a）MapleSim User's Guide：MapleSim的概念信息，MapleSim功能概述、入门教程等。 b）Using MapleSim：关于建模、仿真、分析的帮助主题。 c）MapleSim Component Library：MapleSim中建模元件库的介绍。
MapleSim示例	模型示例，来自不同的工程领域。从Help菜单，选择Examples获取这些示例。
MapleSim用户手册中的示例	用户手册中使用的模型。从Help菜单，选择Examples > User's Guide Examples。这些模型按章节归类。
MapleSim在线资源	网络培训会议、产品演示、视频、应用实例等。 更多信息，见： http://www.maplesoft.com/products/maplesim.
MapleSim模型库	一个免费的MapleSim示例模型库，可以下载示例模型、自定义模型元件、分析模板等，然后应用到您自己的MapleSim项目中。更多信息见： http://www.maplesoft.com/products/maplesim/modelgallery/

第一章 MapleSim 建模环境与多体建模实践

1.1. MapleSim 使用环境

MapleSim 是一个高性能多领域系统建模和仿真工具，建立在符号-数值混合计算技术基础之上，能够有效地处理工程系统模型（例如复杂多领域系统、虚拟设备建模、和控制系统设计）开发中涉及的各种复杂数学问题。MapleSim 提供广泛的预定义模型库和工具箱，包括多体机械库、传动库、液压库、电气库、控制设计工具箱等。研究人员和工程师可以使用广泛的分析工具深入了解他们的系统，同时减少模型开发时间，生成高保真、高性能的模型。

1.1.1 MapleSim 用户界面

MapleSim 使用界面包含下图 1-1 所示的面板和部件：

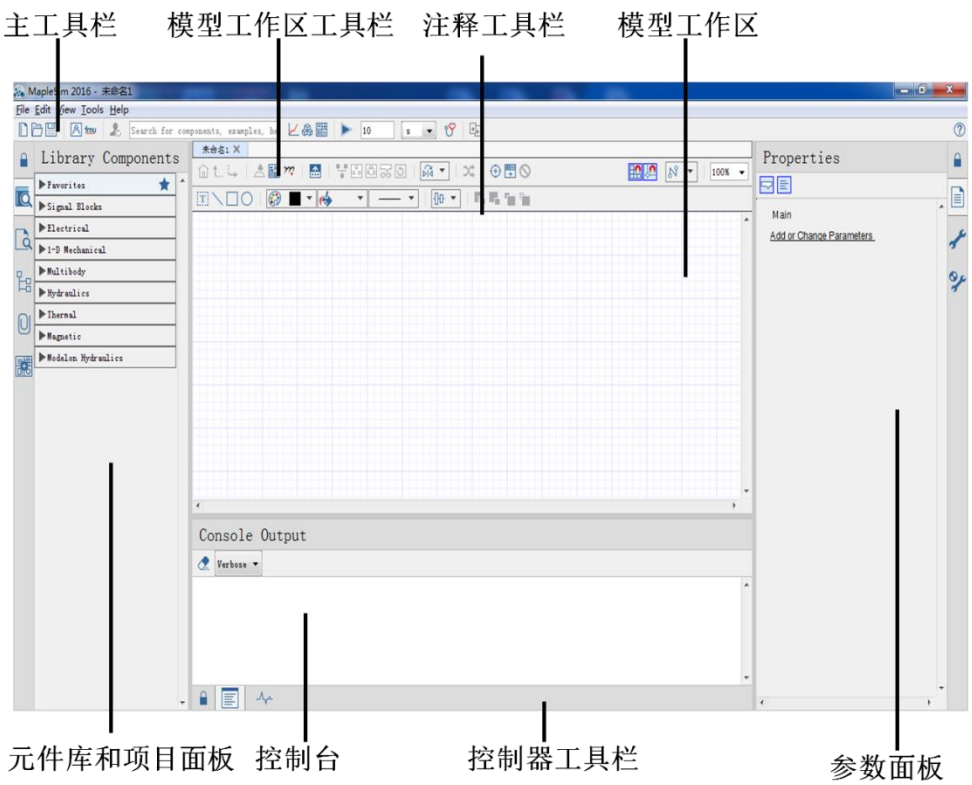


图 1-1　MapleSim 使用界面

表 1-1　**MapleSim 窗口部件功能介绍**

部件	描述
主工具栏	包含的工具有：运行仿真，附加 MapleSim 分析模板到模型中，完成其他常规任务。
控制台工具栏	包含用于选择控制台中显示的消息类型的控件。
模型工作区工具栏	选择对象，布局，添加注释和探针到模型中。
模型工作区	在这个区域，用户可以通过方块图的方式创建和编辑模型。
元件库和项目面板	包含可展开的菜单，可使用其中的模型元件和工具创建模型以及管理 MapleSim 项目。这个面板包含五个选项卡： ● **Library Components** 选项卡：模型库包含构建模型的多领域元件，以及范例模型。 ● **Local Components** 选项卡：包含模型中子系统和自定义组件。 ● **Model Tree** 选项卡：可以看到模型中使用的所有元件。 ● **Attached Files** 选项卡：包含模型的附件，包括文档，参数设定和 CAD 图纸。 ● **Add Apps or Templates** 选项卡：包含用于模型构建和分析的预构建工具。
控制台	控制台：包含下列面板： ● **Console Output** 面板：在模拟期间显示指示 MapleSim 引擎状态的进度消息，并允许您使用图标清除数据 ● **Diagnostics Information** 面板：在构建标识错误所在子系统的模型时，显示用于调试的诊断消息。
注释工具栏	包含添加注释和布局对象的工具。

参数面板	参数面板包含以下选项卡： ● **Properties** 选项卡：允许您查看和编辑建模组件属性，如名称，参数值，初始条件和探针设置。 ● **Simulation Settings** 选项卡：允许用户定义仿真选项，例如仿真时间、求解器的参数值、仿真算法等。 ● **Multibody Settings** 选项卡：让用户定义仿真结果图和图形窗口的布局。 面板中的内容会随着用户在模型工作区中选择的不同对象而变化。

1.1.2 MapleSim 元件库

MapleSim 元件库内置数百个预定义元件，方便您构建模型。这些元件大部分是基于 Modelica® 标准库，依据各自所属的领域分类组织在面板：电子，1-D 和多体机械，热，液压、和信号块等。

表 1-2 MapleSim 元件库

元件库	描述
Electrical	电子元件库，元件包括电子仿真电路、单相和多相系统、机器。
Magnetic	电磁元件库，提供对磁路系统建模的元件。
Hydraulic	液压元件库，包含对液压系统建模的元件，如体动力系统，汽缸，和执行器。
1-D Mechanical	1-D 机械元件库，建模元件包括一维平移和旋转系统。
Multibody	多体机械元件库，提供对多体机械系统建模的元件，包括力、运动、和运动副等。
Signal Blocks	信号块元件库，包括操作和生成输入和输出信号。
Thermal	热元件库，包括热流和热传导等元件。

注：如果您安装了 MapleSim 的其他模块或专业工具箱，MapleSim 左侧的面板会显示相应的面板。

查看元件相关的帮助主题

通过位于模型工作区下方的帮助面板，您可以查看 MapleSim 模型库中元件的帮助主题。下列方式可以打开帮助页：

（1）鼠标右键点击任意面板中的一个建模元件，从弹出的右键菜单中选择帮助。

（2）对于模型工作区中的模型，鼠标单击这个元件，然后按下 F2 键。

（3）可以打开 MapleSim 帮助系统，通过搜索关键词查看元件的帮助页。

1.1.3 运行 MapleSim 内置示例模型

MapleSim 左侧的 examples 面板中收集了一些不同领域的范例模型。示例：从 MapleSim 的主工具栏中，找到 Help > Physical Domains >Multibody> 5 DoF Robot。点击 5 DoF Robot，即可加载模型到工作区域，这是一个 5 自由度的机械臂。

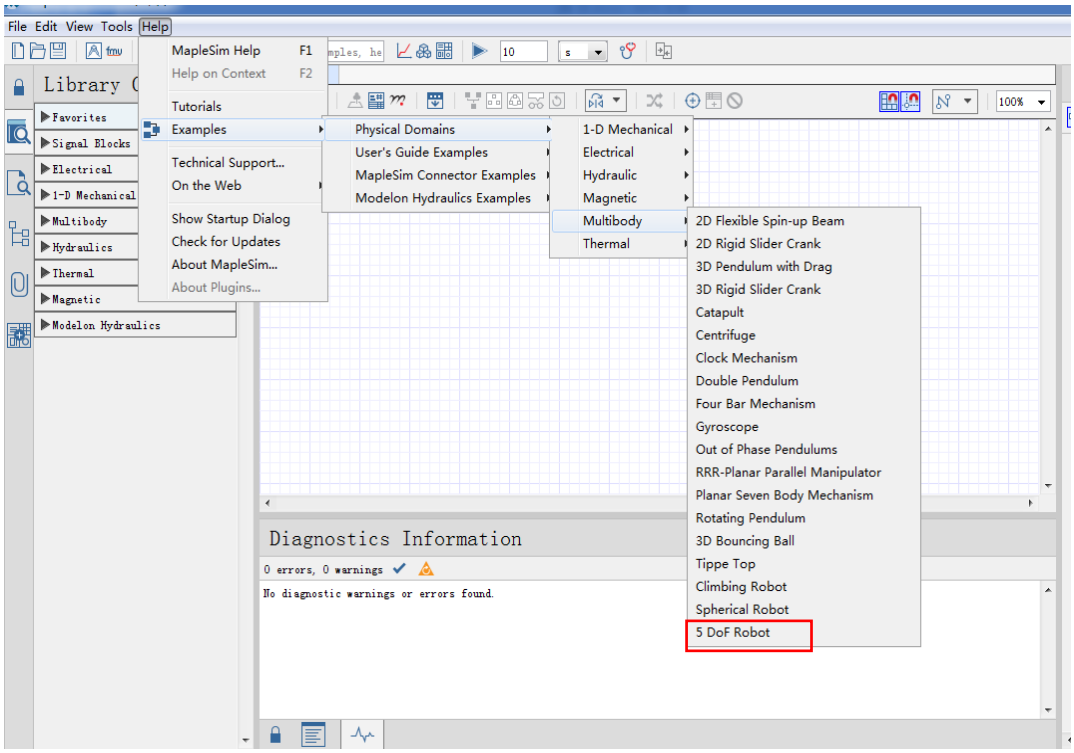


图 1-2　打开 5 自由度的机械臂文件

1.1.4 运行仿真

点击工具栏上图标  ，运行仿真。仿真过程的信息显示在底部的窗口中。但仿真完成后，软件会弹出仿真结果图和 3D 可视化窗口。

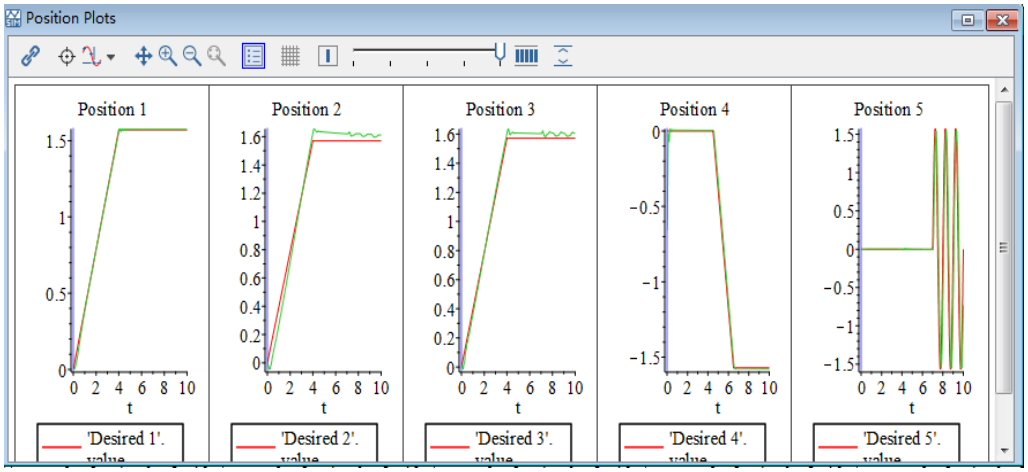


图 1-3　仿真结果图

1.1.5 3D 可视化

按下 3D 可视化窗口上的播放按钮，浏览系统行为的 3D 动画。您也可以自定义动画的视图。

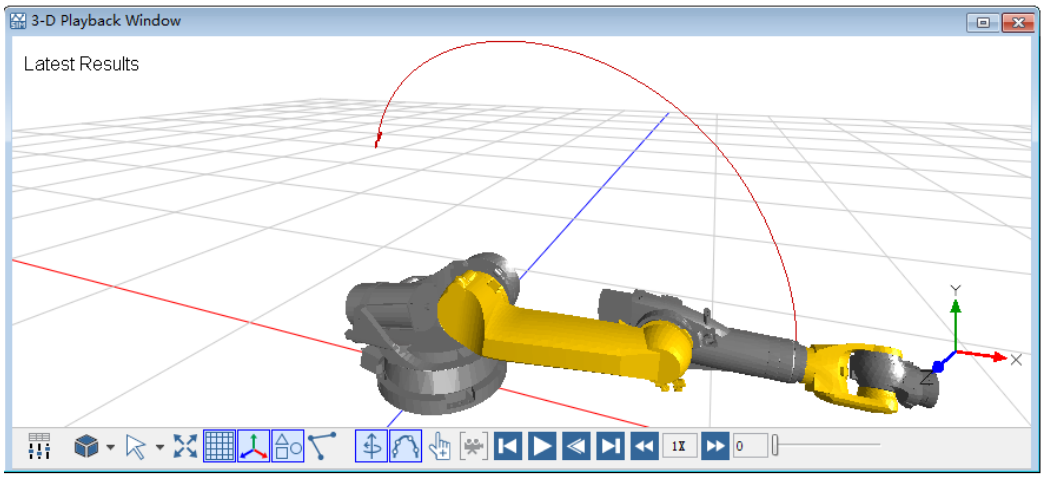


图 1-4　Playback 窗口

旋转动画：按住鼠标左键或右键，然后拖动鼠标指针。平移动画：按住鼠标滑轮或中间的按钮，然后拖动鼠标指针。放大或缩小：按住 Ctrl 键，旋转鼠标滑轮。

1.2. 多体系统建模

本节将介绍 MapleSim 多体机械系统建模的基础知识，下面的第一小节将让用户了解 3D 建模工作环境，第二小节将介绍一些常用的多体建模元件的应用。完成本节后用

户可以仅需完成后面章节的练习，进一步了解 MapleSim 中的多体建模技术。

模型下载地址：http://www.maplesoft.com.cn/book/1.2.zip

1.2.1 三维建模工作区

显示三维模型工作空间：

使用 MapleSim 主工具栏的按钮可以显示三维建模工作空间：

图 1-5　显示三维建模工作空间

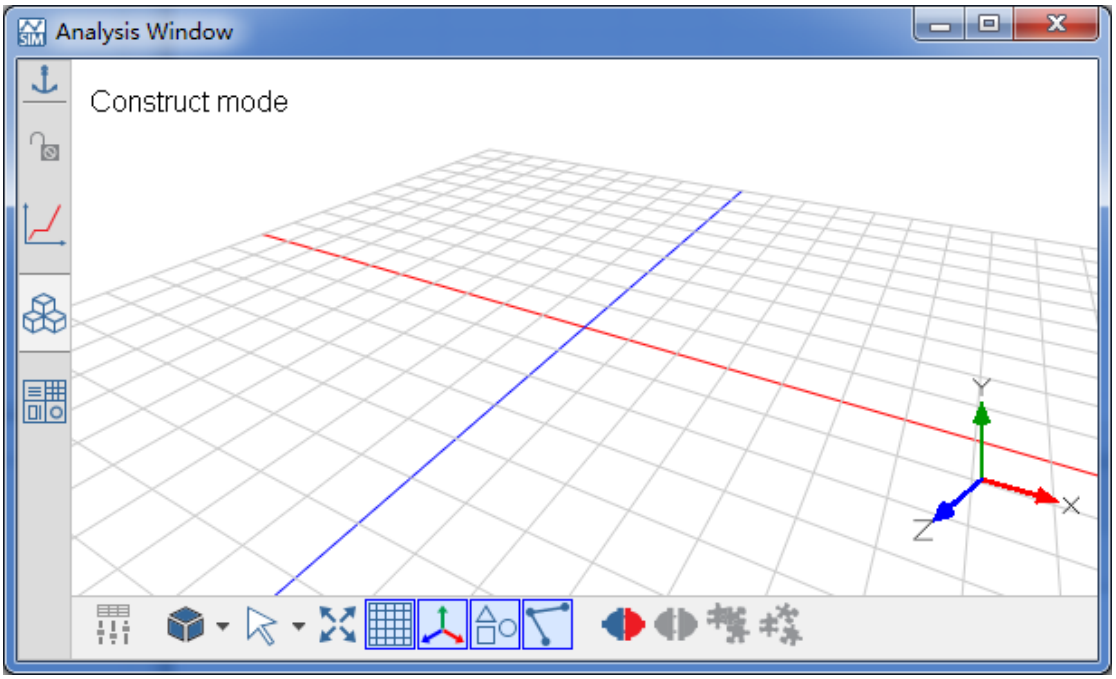


图 1-6　三维建模工作区图

三维建模工作空间有两种模式：Construct 和 Playback。在创建多体机械模型时，Construct 模式可以提供模型的三维预览视图，并且允许用户在三维建模工作空间内直接建模。Playback 模式则用于当建模工作完成之后仿真结果的动画演示。

1．Construct 模式下的主工具栏：

图 1-7　Construct 模式下的主工具栏

2．Playback 模式下的主工具栏：

图 1-8　Playback 模式下的主工具栏

用户可以通过点击图标  中的黑色小三角形在各种不同的视图之间切换（前视，俯视，侧视）。在透视图中，用户可以选择图标  ，从各个角度检查模型。

在 Construct 模式中，用户可以直接把多体元件拖入 2D 或者 3D 的建模区域来创建模型。在 3D 界面中，用户可以通过点击图标  连接两个元件，点击图标  删除连接线。选择图标  ，在无需运行仿真的情况下更新模型的 3D 视图（根据元件的初始条件和设置计算模型的位置）。

在 Playback 模式中，有多个选项可以控制多体模型仿真后的动画。如果模型中有力或者力矩的示意箭头，可以使用图标  来隐藏或显示这些箭头。如果有路径跟踪元件，可使用图标  来隐藏或显示这些线。也可以让动画的镜头跟随指定的元件，用户也可以是点击图标  ，选择模型中特定的元件，让动画的镜头跟随该元件，也就是说让摄像机跟踪该元件在动画中的运动情况，该跟踪功能可以使用图标  激活或者关闭。

参数设置面板

用户可以点击右侧  ，进入“Multibody Settings”栏，并通过修改其中的参数 $\hat{e}_g$ 设置全局坐标系的方向。$\hat{e}_g$ 默认值是[0,-1,0]，表示重力的方向是 Y 轴的负方向，因此在 3D 视图中 Y 轴的正方向是向上的。

1.2.2 常用多体建模元件介绍

刚体和坐标系

Rigid Body - 刚体元件，包含质量和转动惯量的质心系。刚体元件示意图如下，惯量矩阵是相对于局部 x, y, z 坐标系中质心 C 定义的。

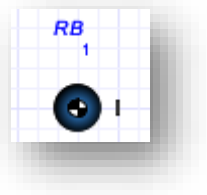


图 1-9　Rigid Body 元件

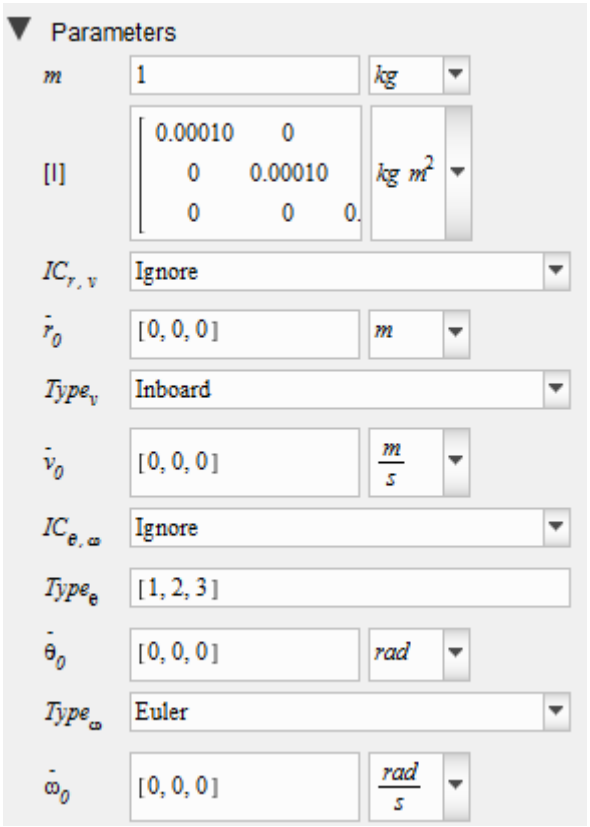


图 1-10　参数设置

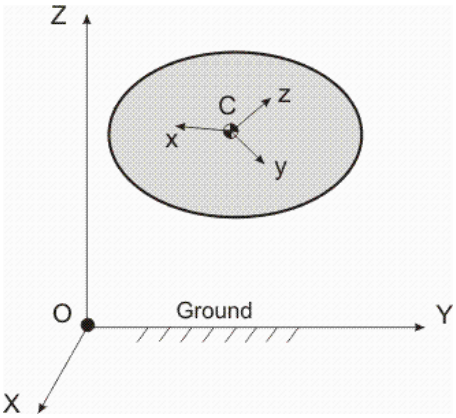


图 1-11　Rigid Body 原理图

Rigid Body Frame – 刚体坐标系元件，是具有相对于刚体元件（Rigid Body）质心系的位移和方向的坐标系。该元件很容易被错认为是机械机构中的连杆，但它不是！它仅是用来作为从 frame a 到 frame b 的坐标转换。因此，一定要注意该元件与哪个元件连接。

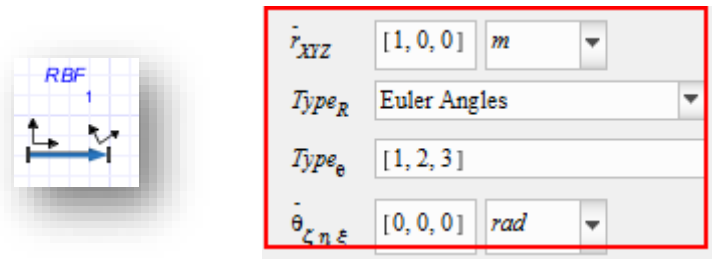


图 1-12　Rigid Body Frame

上图表示在 x 正方向上相对于左侧端口 Frame a 有一个单位长度。请通过下面的模型体会其中微妙的不同之处。

力（驱动）和力矩

多体库中有两种类型的力和力矩元件：“Applied Force/Moment”和“Applied World Force/Moment”。Applied Force/Moment 元件将输入信号解析为作用力/力矩，施加到两个坐标系（Frame）之间。Applied World Force/Moment 元件具有相同的功能，区别在于力/力矩是作用在地面和坐标系（Frame）之间的。

运动副和运动

不同类型的运动副连接刚体坐标系元件（Rigid Body Frame）会产生不同的结果。如果加入了初始条件，会决定运动的方向。同样，请通过下面的模型体会其中微妙的不同之处。

传感器

传感器元件也有两类，分别为相对转动/平移传感器和绝对转动/平移传感器。顾名思义，相对转动/平移传感器测量两个元件之间的相对转动/平移运动，而绝对转动/平移传感器测量元件相对于地面的转动/平移运动。

可视化

除了 MapleSim 多体库中的隐式几何体外，我们也可以通过添加可视化元件来自定义三维模型的外观。MapleSim 多体库中有一些简单的几何体，例如矩形、柱形、球形、锥形等。我们也可以在模型中添加.STL 格式的 CAD 文件。

1.2.3 初始条件控制

我们可以元件的初始条件（IC），这些初始条件可以设置为忽略（Ignore），猜测（Treat as Guess），或者强制（Strictly Enforce）。如果选择忽略（Ignore），MapleSim 将自动计算

模型中刚体（Rigid Body）的初始位置/方向，初始速度为 0。如果选择猜测（Treat as Guess），Maplesim 会以这些初始条件（ICs）为基础寻找一组合理的初始条件配置。最后，用户可以选择强制（Strictly Enforce），强制要求元件遵守定义的初始条件，但需要注意的是如果初始条件中存在不一致或前后矛盾的初始条件，那么会导致模型初始化和仿真失败。例如在创建模型时，用户可以使用强制（Strictly Enforce）设定模型的参考点（直角坐标），然后用 Rigid Body Frame 定义其他元件的相对位置，从而快速准确地构建模型。

1.2.4 创建一个单摆模型

使用多体库中的元件构建一个单连杆单摆。

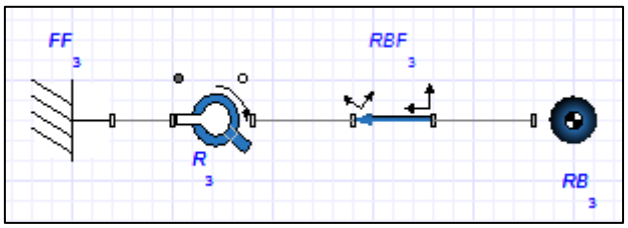


图 1-13 单连杆单摆模型

首先从 MapleSim 界面左侧面板 Libraries 下展开 Multibody 库。

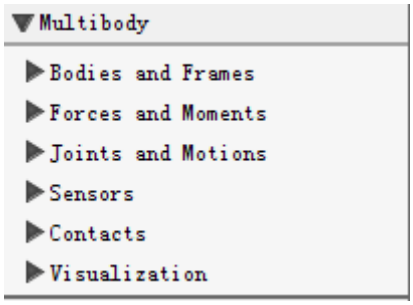
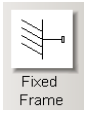
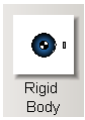


图 1-14 Libraries 面板中 Multibody 库

找到下面表格中的元件，然后用鼠标左键将这些元件拖入到模型工作区 Model WorkSpace。

表 1-3 单摆模型中的元件

元 件 数量	元件名称和位置	图形
1	Multibody > Bodies and Frames > Fixed Frames	
1	Multibody > Bodies and Frames > Rigid Body	
1	Multibody > Bodies and Frames > Rigid Body Frame	
1	Multibody > Joints and Motions > Revolute	

提示：一些元件可能需要旋转或翻转。操作方式是鼠标右击该元件，然后从弹出的菜单中选择需要的操作。

在元件端口的上方移动鼠标，直到出现一个绿色的点，鼠标左键点击一下该端口开始连接。移动连接线到另一个元件的端口，点击该端口完成连接。

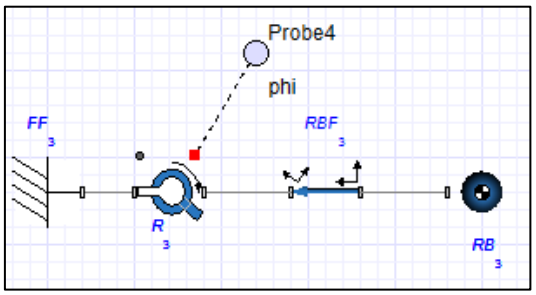
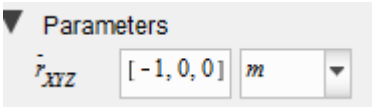


图 1-14　连接后的模型

鼠标右击旋转副（revolution joint）右上角的端口，附加一个探针（probe）。从右侧 properties 面板中选中 Angle，将 phi 修改为 Angle。

创建上图显示的模型后，修改下列参数值。选中模型中的元件，MapleSim 界面右侧的 properties 面板中出现该元件关联的所有参数，用户可以修改这些默认的参数值。

表 1-4 元件参数修改

元件	参数改变
第一个刚体坐标系 	 上面的参数值将连杆的长度设置为距离质心 x 轴反方向 1m。

仿真

点击工具栏上图标  ，运行仿真。仿真过程的信息显示在底部的窗口中。但仿真完成后，软件会弹出仿真结果图和 3D 可视化窗口。

对输出的仿真结果图，用户可以像 Maple 中创建的任意图形一样操作。鼠标右键点击图形，从弹出的右键菜单中选择期望的图形操作。

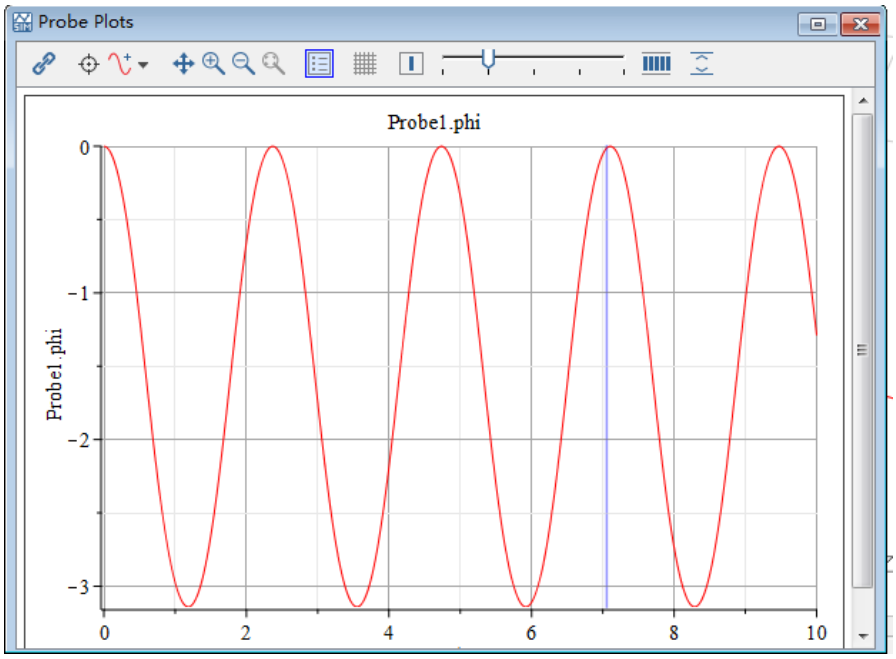


图 1-15　仿真结果图

仿真结果的三维可视化

按下 3D 建模窗口上的播放按钮  ，浏览系统行为的 3D 动画。

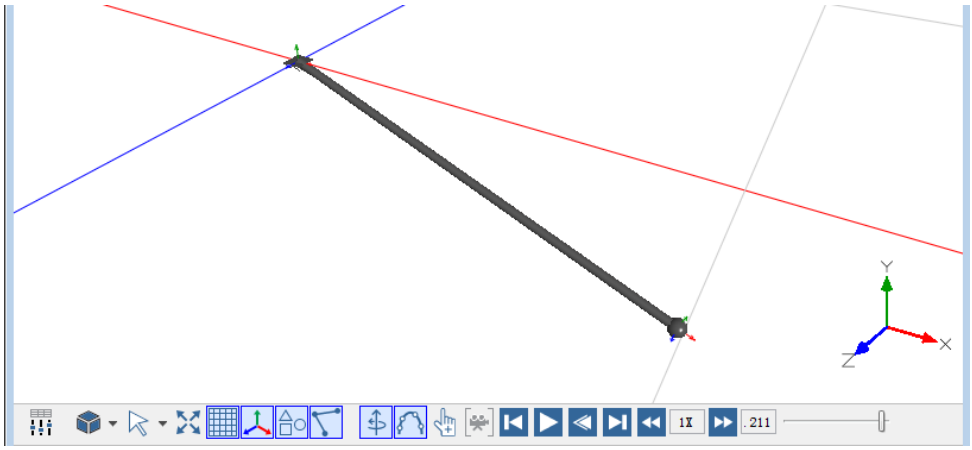


图 1-16　三维工作空间内的动画显示

1.2.5 添加一个基于数学方程的自定义元件

现在通过创建一个自定义元件，在模型中添加摩擦力。

1. 点击左侧项目面板上的 Add Apps or Templates 选项卡（  ）
2. 从列表中选择 Custom Component。
3. 在窗口底部的 Attachment 栏内将文件名修改为 Friction。

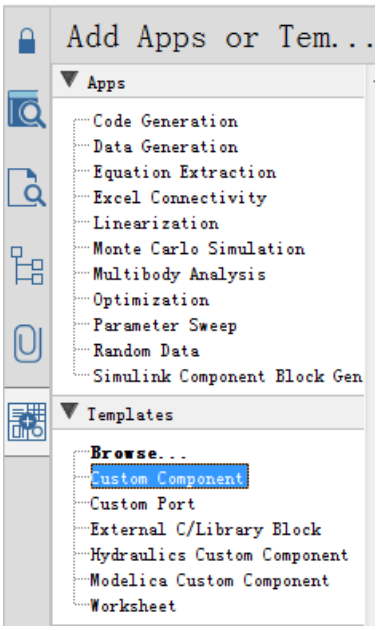


图 1-17　选项卡中的 Custom Component

4.点击  。MapleSim 会自动进入 Maple 窗口 DAE Custom Component Template

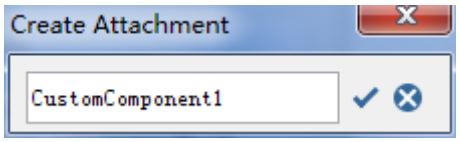


图 1-18　输入自定义元件的名称

提示：可在左侧 Project 面板下的 Attachments→Attachments→Custome Components 找到该文件。

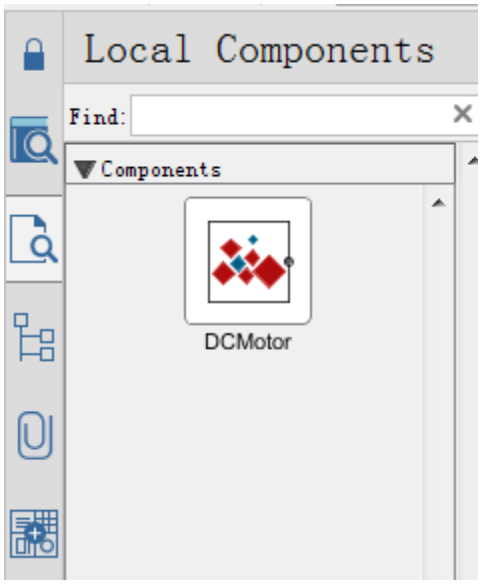


图 1-19　Local Components 面板中的自定义元件

5. 在 Custom Component Template 创建自定义元件。

在 Define Equations 段落，输入定义元件属性的数学方程 eq。

在 configuration 段落，输入定义建模元件属性任意的参数和初始值，以及图标接口。

在 Component Generation 段落，修改元件的名称为 friction。

Define Equations:

$eq := [\mathrm{tau}(t) = f \cdot diff(\mathrm{theta}(t), t)]$

$$\left[\tau(t) = f\left(\frac{d}{dt}\theta(t)\right)\right]$$

图 1-20　输入函数方程

希腊字母的输入：

方法 1：直接输入希腊字母对应的英文，theta(t)，tau(t)。

方法 2：使用 Maple 左侧“希腊字母”面板中的字母，输入 θ (theta)和 τ (tau)。

方法 3：使用 Maple 左侧“手写识别”输入希腊字母。

导数符号的输入：

方法 1：使用 diff(theta(t),t)，详细介绍参考 diff 的帮助。

方法 2：输入 θ，按下“Ctrl+Shift+”组合键，光标会移到 θ 的上方，然后使用句号键“.”插入点符号，数学意义是 diff(theta(t),t)。

按下 Add Port 键，点击该端口，按住鼠标左键将它拖到元件的底部位置。在 configuration>Ports 中，按下 Clear All Ports 键清除模板中的所有预定义端口。

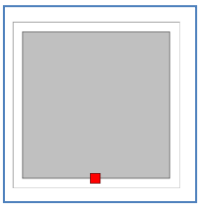


图 1-21　自定义元件方块图

在 type 中选择 Rotational。

Angle 部分，从下拉菜单中选择 theta(t)。

Torque 部分，从下拉菜单中选择 tau(t)。

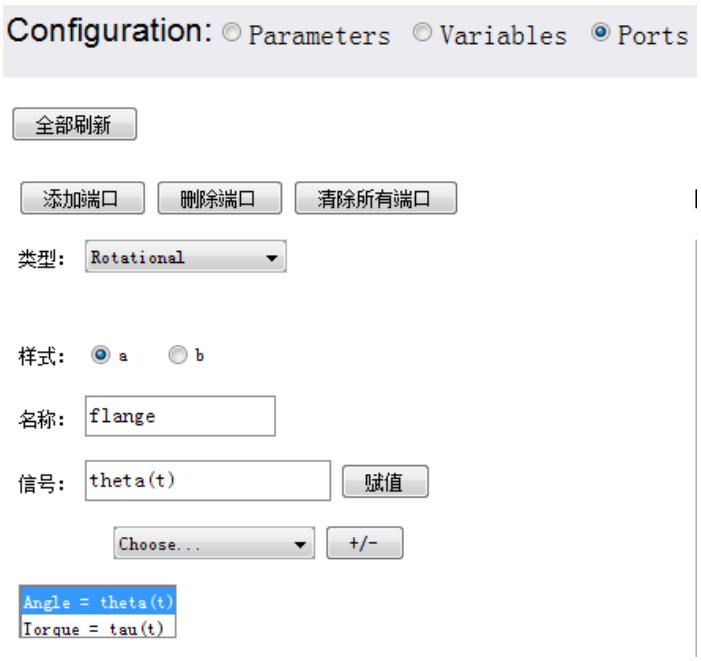


图 1-22　自定义元件端口定义

按下生成 MapleSim 元件按钮创建自定义元件块，生成的自定义元件的同时将自动返回到 MapleSim 环境。创建的自定义元件出现在 Local Component 选项卡内的 Component 中。

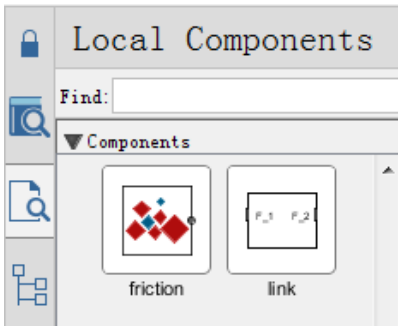


图 1-23　Local Components 面板中的自定义元件

将自定义元件 friction 拖入到模型区域，然后将它连接到铰链的右上角。

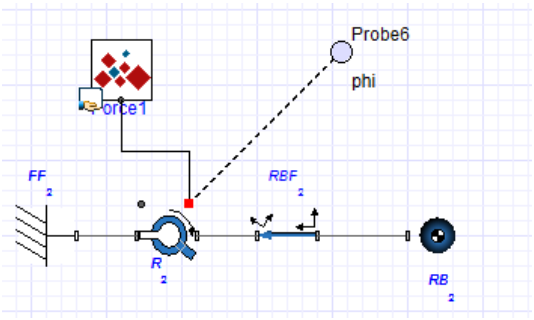


图 1-24　加入自定义元件后的模型

选中自定义元件 friction，在右侧 Properties 中，修改参数 f 为 0.5。

点击工具栏上的按钮  ，运行仿真。用户可以通过动画和结果图形发现摩擦力元件对系统的影响。

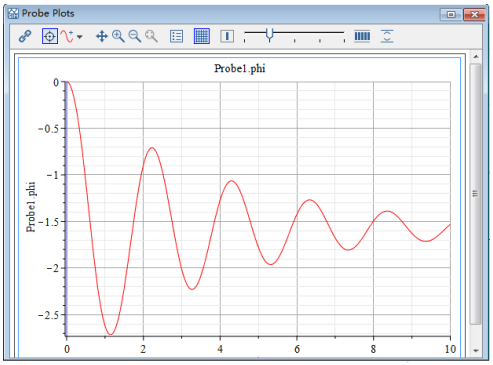
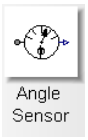
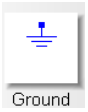


图 1-25　仿真模型和结果图

1.2.6 添加控制器到单摆模型

添加表 1-5 中的元件到模型中，创建一个由控制电机和单摆组成的多领域模型。

表 1-5 控制器建模元件表

元件数量	元件名称和位置	视图
1	1-D Mechanical > Rotational > Sensors> Angle Sensor	
1	Electrical > Analog > Common > Ground	
1	Electrical > Analog > Sources > Voltage > Signal Voltage	
1	Electrical > Machines > DC Machines > DC Permanent Magnet	
1	Signal Blocks > Common > Constant	
1	Signal Blocks > Common > Gain	
1	Signal Blocks > Common > Feedback	

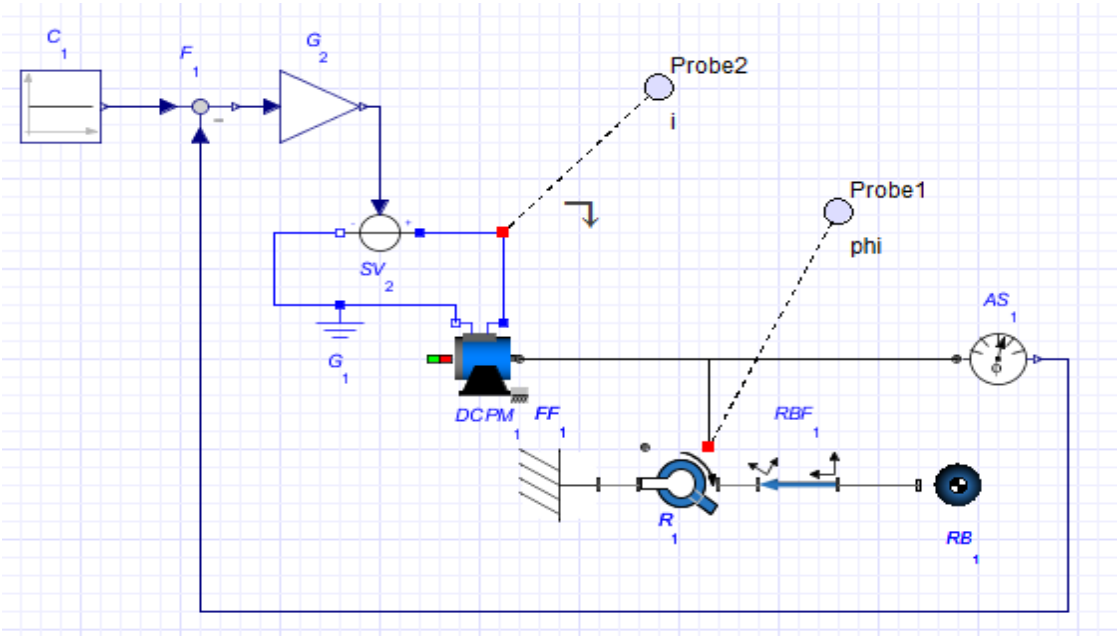


图 1-26 受控单摆模型

鼠标右击电压和 DC motor 之间的连线，添加一个测量电流的探针。

完成如上图所示的模型后，按照表 1-6 修改参数：

表 1-6 元件参数修改

元件	参数修改
Constant Signal 	k=0 **这个控制器参数的目的是保持连杆 x 轴和 z 轴之间的角度，使其尽可能接近 0。**

创建子系统

通过创建子系统，可以更好地从视觉和功能上管理模型。现在我们要创建一个机械臂子系统。

按住鼠标左键，在 revolute，rigid body，和 rigid body frame 元件周围拖出一个方框。

点击鼠标右键，从右键菜单中选择 Create Subsystem.

按下组合键 Ctrl+G 创建一个子系统

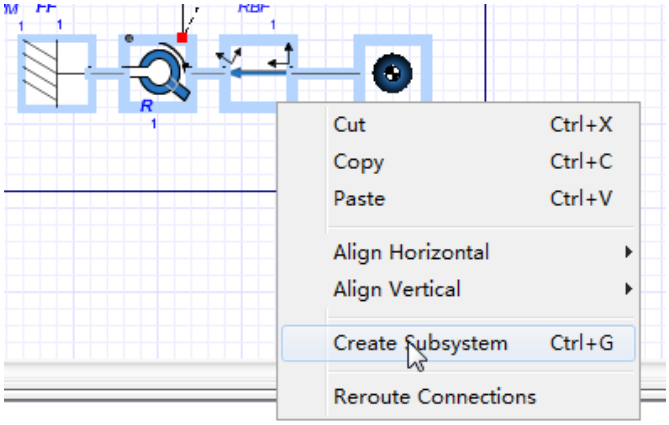


图 1-27　创建子系统

将子系统命名为 Arm。按回车键确认。用户可以通过双击该子系统或工具栏上导航按钮查看子系统。

1.2.7 自定义仿真结果图

首先点击创建一个新的图形窗口显示电流与角度的对比图。操作方式是，点击 Show Simulation results（  ）查看仿真结果，点击 Simulation Results（）显示左侧的面板，在 Plot Window 中，点击（  ）复制，并在对话框内输入图形窗口名称“电流 Vs 角度”。

打开角度 phi 随时间 t 变化的图，在面板的第二栏 Variables 中对图中的变量进行修改，点击电流 i，右键选择 change the X-aris，则图形中的横坐标会从原本的时间 t 变化为电流 i，从而得到电流-角度图。

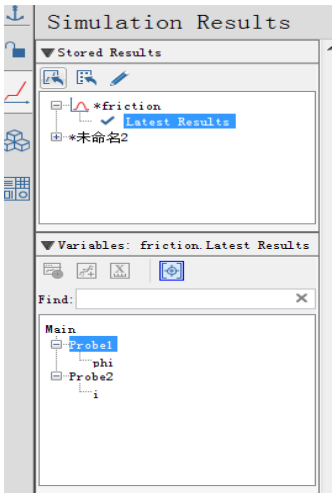


图 1-28　创建自定义图形窗口

X-axis 选择 i（电流），Y-axis 选择 phi（角度）。运行仿真，显示如下图所示的结果图。

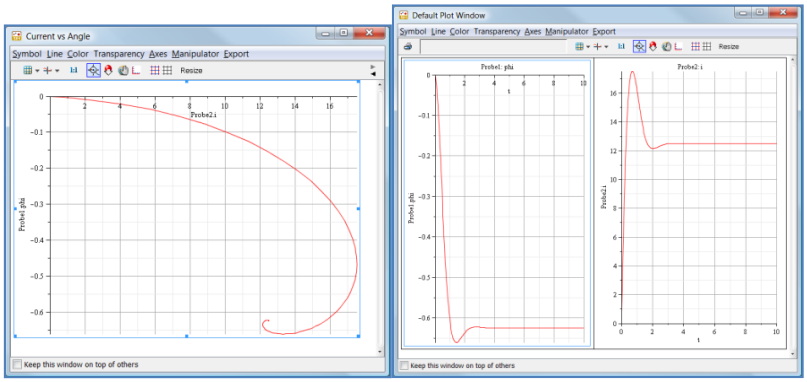


图 1-29　仿真结果图

管理结果

点击 Analysis Window 界面左侧的 plot window 中的（  ）按钮，可以将所选图形结果保存。

在仿真结果窗口中选择需要保存的结果，如下图所示，打勾的结果会被导出。然后单击导出数据按钮，将弹出保存结果的对话框。

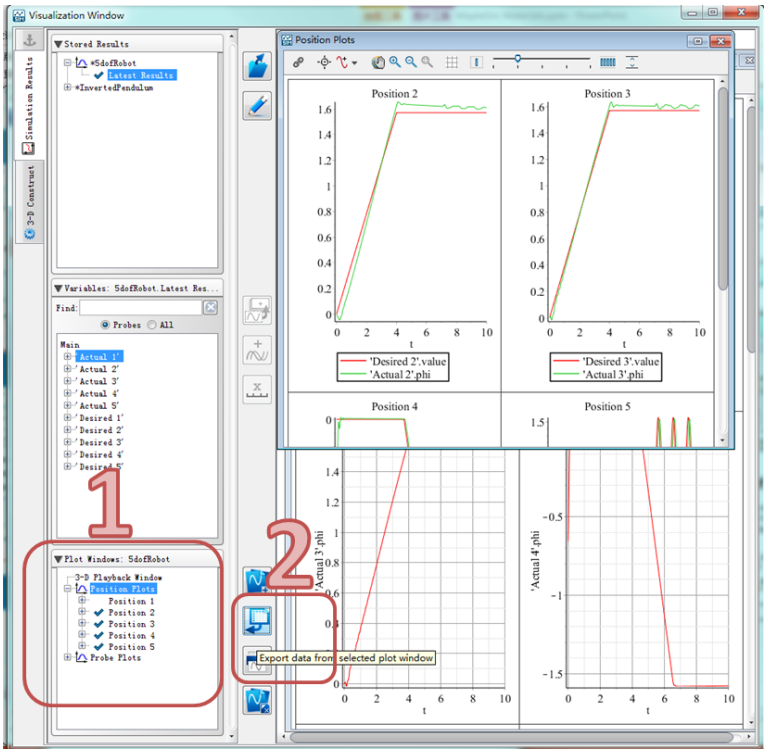


图 1-30　仿真结果保存

给文件取名及设置保存地址，可以将仿真结果输出为 Excel 或 csv 格式的文件。

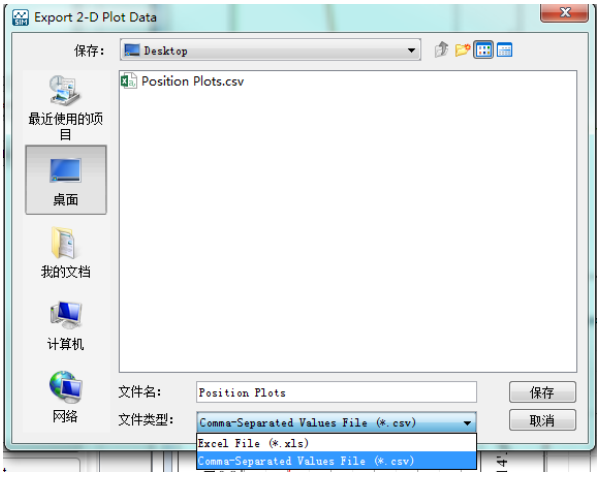


图 1-31　保存结果

1.2.8 使用 Maple 模板分析 MapleSim 模型

MapleSim 基于 Maple 引擎，与 Maple 紧密集成，因此用户可以使用 Maple 的命令、图形工具、内嵌元件、技术文件等分析和操作 MapleSim 模型或子系统的动力学行为。例如，用户可以使用 Maple 提取和操作模型的方程，测试输入和输出值，转换模型为 C 代码，完成大量的高级分析任务。

用户可以使用 MapleSim 模板对话框中的预置任务模板在 Maple 中处理模型。这些模板是 Maple 工作表，通过预定义功能的按钮完成建模和分析任务：首先创建一个 MapleSim 模型，然后打开一个可用的模板在 Maple 中完成分析任务。

MapleSim 提供表 1-7 所示的预置任务模板：

表 1-7　MapleSim 内置任务模板

模版名称	界面显示名称	目的
C 代码生成模板	Code Generation	转换模型到 C 代码。
自定义元件模板	Custom Component	创建一个基于数学模型的自定义建模元件。更多信息，请参考第四章“创建自定义建模元件”。
数据生成模板	Data Generation	定义和生成一个使用在 MapleSim 中的数据文件，例如用于插值表的数据。

方程提取模板	Equations	从线性或非线性模型中提取方程。
离散状态空间自定义元件	Custom Discrete State Space	定义和生成一个由离散状态空间描述的自定义 MapleSim 建模元件。
离散传递函数自定义元件	Custom Discrete Transfer Function	定义和生成一个由离散传递函数的自定义 MapleSim 建模元件。
Excel 连接模板	Excel Connectivity	导入 Excel 电子表格作为 MapleSim 参数集，或者输出 MapleSim 参数集到 Excel 电子表格中。
线性系统分析模板	Analysis	查看和分析线性系统的方程。
线性化	Linearization	从 MapleSim 中的连续子系统模型创建一个线性系统对象。
自定义元件模板：Modelica 代码定义	Modelica Custom Component	使用 Modelica 代码定义和生成一个 MapleSim 自定义元件。
蒙特卡洛模拟模板	Monte-Carlo Simulation	定义一个参数的随机分布，并使用随机分布参数对模型仿真。
多体机构分析模板	Multibody Analysis	提取多体系统的方程，并显示为一种适宜操作和分析的格式。
参数优化模板	Optimization	分析和编辑模型的参数，查看可能的仿真结果。
随机数据模板	Random Data	定义和生成一组随机数据点，例如，生成一组用于插值表元件的数据集。
灵敏度分析	Sensitivity Analysis	完成参数的灵敏度分析。
工作表	Worksheet	创建一个连接 MapleSim 模型的工作表。

另一个编辑和分析模型的方式是，用户可以插入一个 MapleSim 模型内嵌元件到 Maple 工作表中，在该元件中打开一个已有的 MapleSim 模型。

上述两种完成分析任务的方式，都让用户可以使用任意的 Maple 函数包中的命令通过编程方式对模型工作，包括 MapleSim 和 DynamicSystems 函数包。

注意：当使用这些文件模版后，保存.mw 文件，然后保存对应的.msim 模型文件。

现在使用 Templates Button 中的 Monte-Carlo 模板对模型进行研究。

点击左侧工具栏上的  按钮，从菜单中选择 Monte-Carlo Simulation。

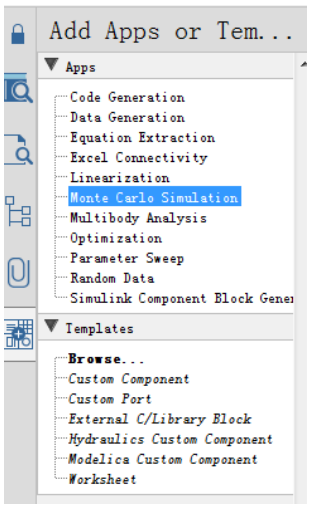


图 1-32　Add Apps or Template 面板

双击打开 Mote Carlo Simulation。

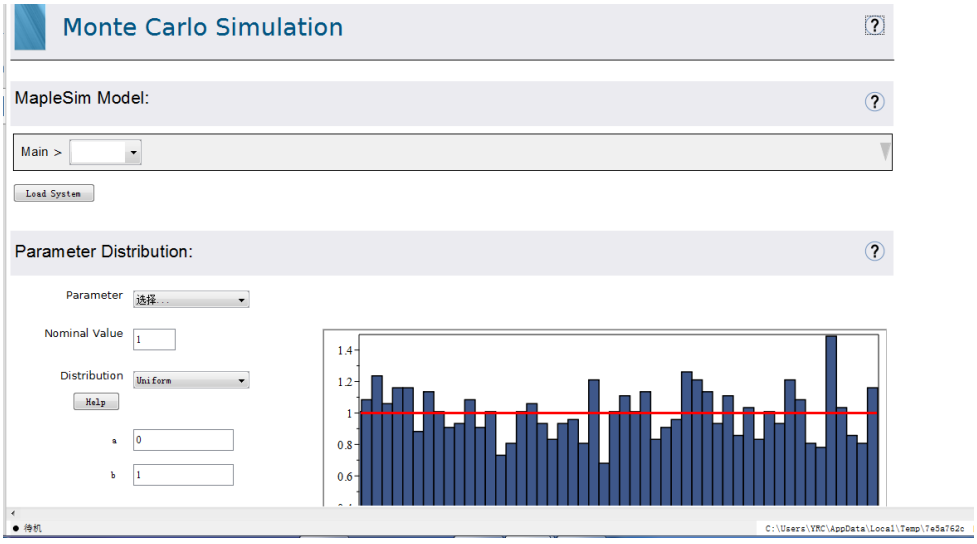


图 1-33　蒙特卡洛模拟模板

首先在 Monte-Carlo Simulation 段落，点击 Load System 按钮。

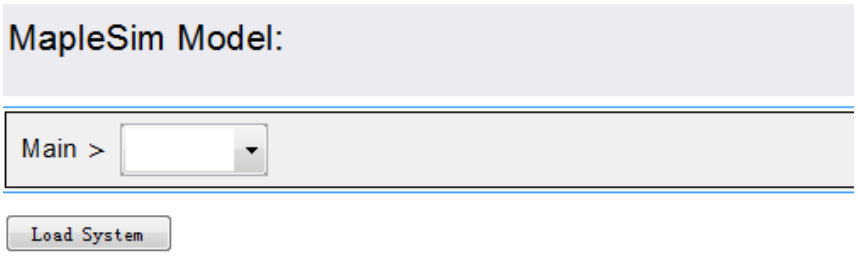


图 1-34　提取系统参数窗口

然后设置如下的仿真条件：

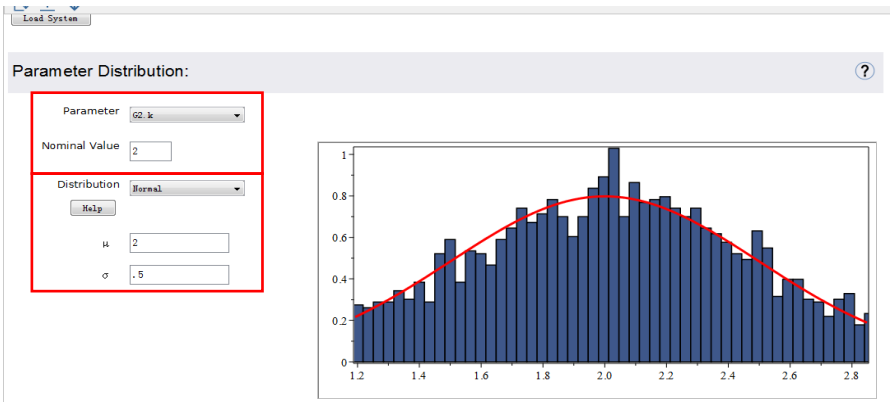


图 1-35　选取参数和分布类型

将参数 all variables 修改为 Main.probe.phi。按下 Run Simulation 按钮运行仿真。

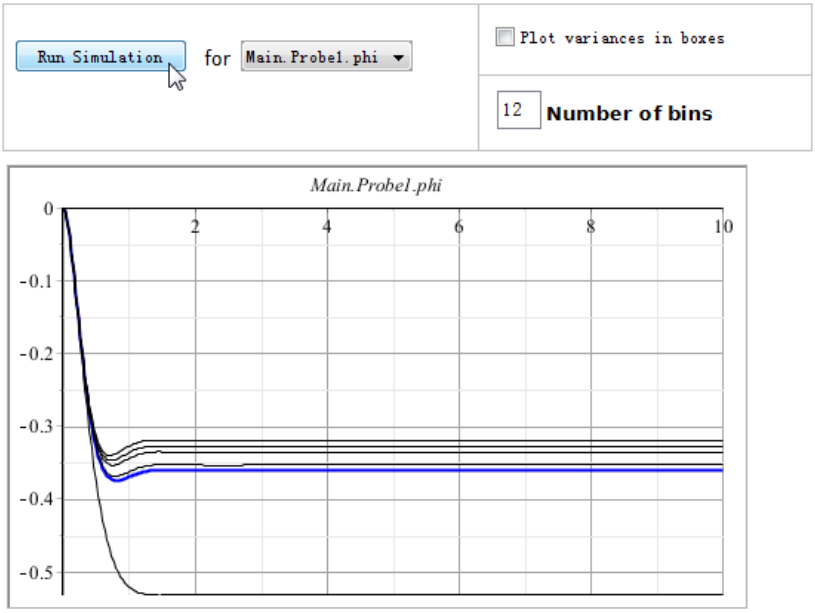


图 1-36　蒙特卡洛模拟结果

1.3 曲柄滑块模型

模型下载地址：http://www.maplesoft.com.cn/book/1.3.zip

在这个练习中，用户将使用多体机械库中的元件，对平面曲柄滑块机构进行仿真，原理图如下：

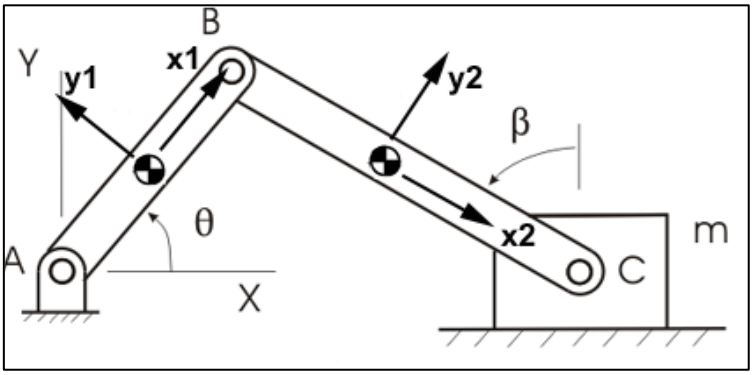


图 1-37　平面曲柄滑块机构原理图

这个模型包含 3 个旋转副：第一个旋转副（A）连接在平面连杆的一端，平面连杆的另一端与连接杆上的第二个旋转副（B）连接；连杆的另一端与滑动质量块上的第三个旋转副（C）连接；滑动质量块通过平移运动副与地面连接。实际上，这个机构被用于将曲柄的旋转运动转换为滑动质量块上的平移运动，反之亦然。上图是机构的概念图，质量是唯一的外力，方向是沿着 Y 负方向。

在练习中，我们将完成下列任务：

1. 创建一个平面连杆子系统；
2. 定义和分配子系统参数；
3. 创建曲柄和连杆元件；
4. 在模型中添加固定架，滑动质量，并连接元件；
5. 定义初始条件；
6. 仿真平面曲柄滑块机构。

1.3.1 创建一个连杆子系统

从原理图中，用户可以看到曲柄－滑块有两个关联的平面连杆：曲柄（从 A 到 B 的连杆）和连杆（从 B 到 C 的连杆）。这两个连杆在局部坐标系中 x 轴上都有分量（分别

是 x_1 和 x_2）。因此，用户将首先创建一个通用的平面连杆，连接两个端口。内侧端口（基座）位于沿着连杆 x 轴的 $-\frac{L}{2}$ 单位上，同时外侧端口（尖端）位于沿着连杆 x 轴的 $\frac{L}{2}$ 单位上。在这个示例中，L 指连杆的长度，质心假设位于连杆的中心。

1）打开一个新的 MapleSim 文件。

2）从 Multibody → Bodies and Frames 菜单，添加两个 Rigid Body Frame 元件和一个 Rigid Body 元件。

3）在模型工作区，鼠标右击 RBF_1 元件，然后选择水平旋转 Flip Horizontal。

4）鼠标右击 Rigid Body 刚体元件，然后选择逆时针旋转 Rotate Counter Clockwise。

5）拖动元件如下图所示的位置。

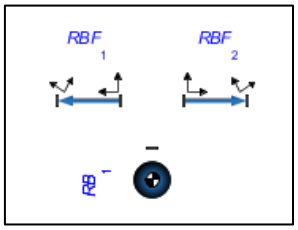


图 1-38　连杆建模元件

用户现在可以连接各个元件，将各个元件有效地组合。

当连接 Rigid Body Frame 组件到 Rigid Body 组件时，需要连接 Rigid Body Frame 的内侧端口（也就是有交叉斜线的圆圈）到 Rigid Body 元件的质心。保证局部参考坐标系用于描述 Rigid Body Frame 元件的位移和旋转，与定义在 Rigid Body 元件质心上的参考坐标系相匹配。

6）在 Rigid Body 元件和 RBF_1 元件右侧框之间拖出一个连接线。

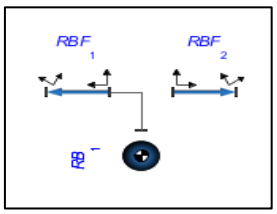


图 1-39　连杆模型

7）在 Rigid Body 元件和 RBF_2 元件左侧框之间建立第二个连接线。

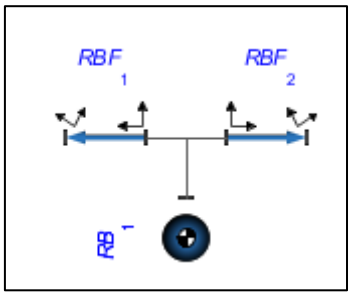


图 1-40　连杆模型

8）使用选择工具（ ），在元件周围拖出一个方框。

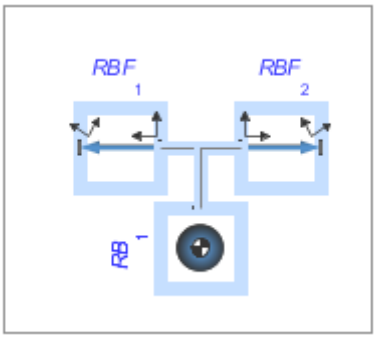


图 1-41　创建子系统

9）从 Edit 菜单，选择 Create Subsystem。

10）在 Create Subsystem 对话框，输入 Link 并点击 OK，用户现在将添加端口，把其他元件连接到这个子系统。

11）双击 Link 子系统。

12）点击 RBF_1 元件的左端口，然后点击子系统的左边框，创建一个端口，如图所示。

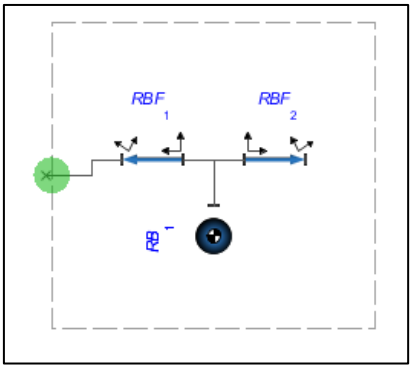


图 1-42　添加子系统端口

13）点击一下连线，一个子系统端口就被添加了。

14）同样方式，使用 RBF_2 元件的右侧框创建另一个端口，如图所示。

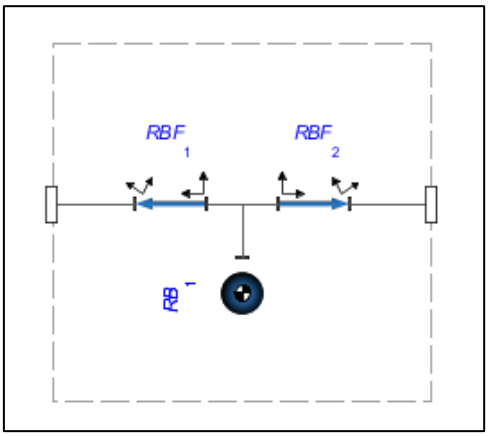


图 1-43 添加子系统端口

1.3.2 定义子系统参数

在这个建模任务中，将定义一个子系统参数 L 表示 link 的长度，并将参数值作为一个变量分配给 Rigid Body Frame 元件的参数。Rigid Body Frame 元件将继承参数 L 的数值，参数 L 的作用等同于子系统的局部变量。用户可以为用户的模型设置任意的参数值，这里将分别实验不同的长度值。

1）从工作区下方的 Libraries 视图中选择子系统中的一个元件，点击模型工作区工具栏上的按钮，切换到参数编辑器视图。

2）在 Link subsystem default settings 表格的第一行，定义一个名为 L 的参数，并按回车键确认。

Link subsystem default settings

Name	Type	Default Value	Default Units	Description
L	Real	1		Length
m	Real	1		mass

图 1-44 定义子系统参数

3）定义一个默认值 1，并输入 Length 作为描述。

4）点击工具栏上的按钮，切换到模型视图。

5）在右侧的 Properties 面板中修改 Parameters，定义下面的参数值：

✧ 对于 RBF_1 元件，在 $\bar{r}_{XYZ}$ 区域定义一个偏移量$[-\frac{L}{2},0,0]$。

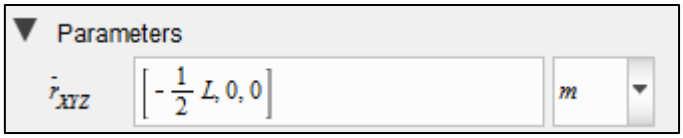


图 1-45　定义 RBF_1 元件的参数

✧ 对于 RBF_2 元件，在 $\bar{r}_{XYZ}$ 区域定义一个偏移量$[\frac{L}{2},0,0]$。

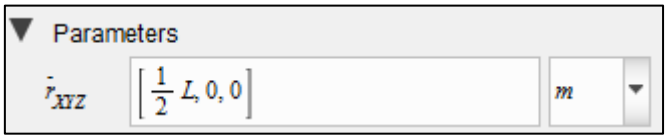


图 1-45　定义 RBF_2 元件的参数

备注：输入分数时使用斜杠键（/）。

6）相似的操作，在 Link subsystem default settings 中添加质量参数 m，默认值为 1，并输入 mass 作为描述。

7）点击工具栏上的按钮，切换到模型视图。

8）选中质量元件，在右侧的参数面板中定义质量矩阵为：

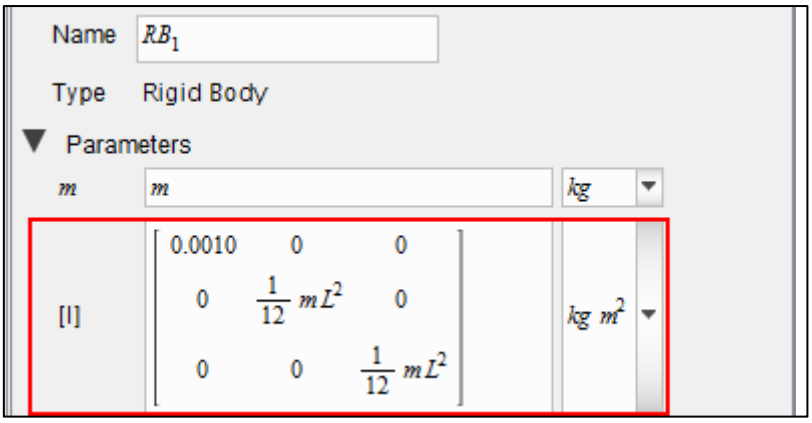


图 1-46　定义 Rigid Body 元件的参数

1.3.3 创建曲柄和连杆元件

在这个任务中，为了创建曲柄和连接杆元件，用户将添加另一个 Link 子系统实例到模型中，并重新命名。重新为连杆元件分配一个不同的长度值。

1）点击导航工具栏上的 Main 按钮，浏览模型的顶层。

2）鼠标右击 Link 子系统，选择 Convert to shared Subsystem。一个 Link 子系统被添加到 Local Component 选项卡中的 Component 面板内，转换为一个共享子系统。

3）选择 Link 共享子系统。

4）在右侧 Properties 选项卡的 Name 区域，修改共享子系统的名称为 Crank。

5）从 Definitions 面板中，将 Link 图标拖入到模型工作区，放在 Crank 子系统的右边。

6）选择 Link 共享子系统的第二个 copy。

7）在右侧 Properties 选项卡，将子系统的名称修改为 ConnectingRod，修改长度值为 2。

1.3.4 添加固定坐标系、滑动质量、和运动副元件

在这个建模练习中，用户将添加一个 Fixed Frame 元件，一个代表滑动质量块的 Rigid Body 元件，以及运动副元件。

1）从 Multibody → Bodies and Frames 面板拖入一个 Fixed Frame 元件，放在 Crank 子系统的左边。

2）从相同的面板中拖入一个 Rigid Body 元件，放在略低于 Connecting Rod 子系统的右下侧位置。

3）添加下面的连接：

✧ 从 Multibody → Joints and Motions 面板拖入一个 Revolute 旋转副，放在 Fixed Frame 元件和曲柄之间，在曲柄和连杆之间放入第二个 Revolute 旋转副，在连杆和滑动质量块之间放入第三个 Revolute 旋转副。

✧ 从同一面板中添加一个 Prismatic 单自由度移动副，放在 Crank 子系统的下方。

4）选择模型工作区内的 RB_1 元件，重新命名为 Sliding Mass。

5）鼠标右击 Sliding Mass 元件选择 Flip Horizontal；用相同的方式调整旋转副 R_3 的方向。

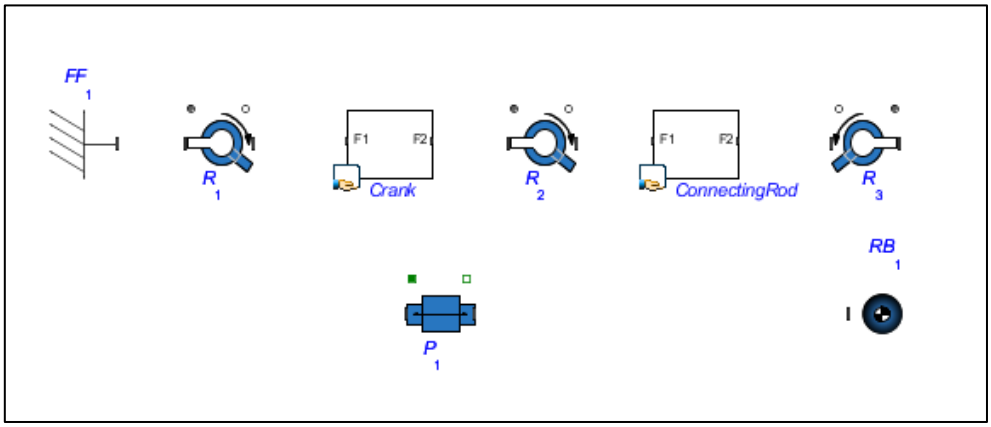


图 1-47　添加元件到模型工作区

6）如下图所示连接各个建模元件。

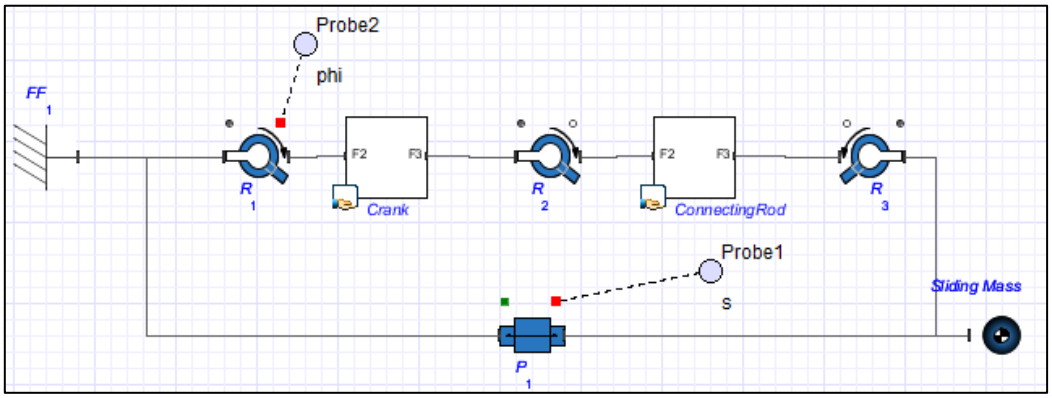


图 1-48　曲柄-滑块模型

提示：在这个示例中，旋转和棱柱节点排成一行。例如，节点的旋转假设相对于期望的坐标系。例如，旋转节点假设围绕内侧框架的 Z 轴旋转，与 XY 平面系统中的 Z 轴一致。滑块与固定节点当创建非平面模型时，这些坐标轴可能需要改变，以确保他们沿着或围绕着正确的方向运动。

1.3.5 定义初始条件

为了定义初始条件，用户需要为模型中一些元件设置参数值。

1）第一个旋转运动副，在 θ_0 区域，设置运动副的初始角为 $\frac{\pi}{4}$ rad。

备注：输入 π 时，键入 Pi，按下 ESC 键，从下拉菜单选择 π 符号。

2）从 $IC_{\theta,\omega}$ 下拉菜单，选择 Strictly Enforce。

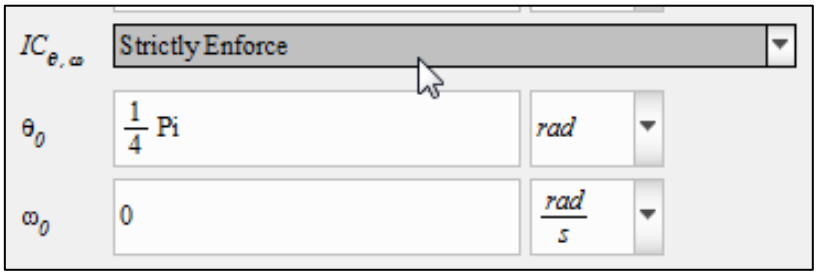


图 1-49　定义初始条件

当 MapleSim 求解器获取初始条件时，首先设置第一个角为 $\frac{\pi}{4}$ rad，然后再设置其他运动副的角度。

1.3.6 仿真

1）从模型工作区的工具栏，点击探针按钮 。

2）在模型工作区，点击 Prismatic 运动副图标右上方，添加一个探针。

3）点击模型工作区中的探针。

4）在 Properties 选项卡中，选择 Length 量测量位移。

5）用相同的方式，在 R_1 运动副图标右上方，添加一个新的探针测量 Angle 量。

6）点击模型工作区的空白区域。

7）在右侧 Simulation Settings 选项卡上，设置 t_d 的参数值为 10 秒。

8）点击工具栏上仿真按钮 。仿真结束后，将显示下面的图形窗口。

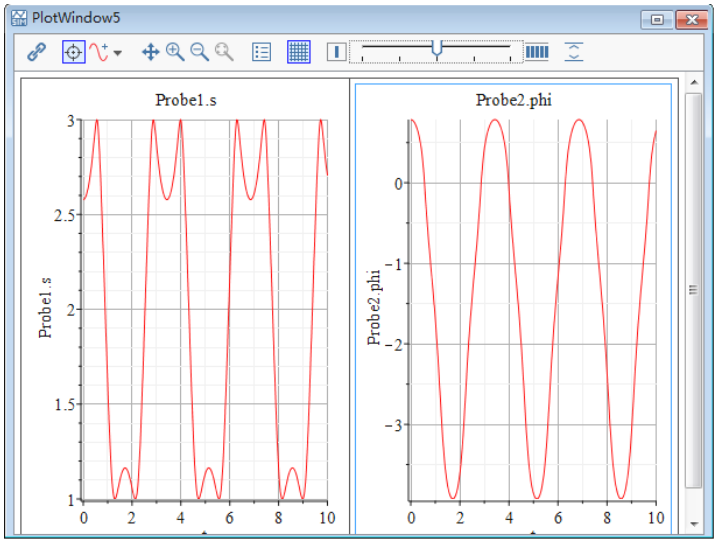


图 1-50　仿真结果图

9）保存模型到“曲柄滑块.msim”。

1.3.7 提取模型的符号方程

现在使用 Templates 中的 Worksheet 模板对模型进行分析。

按下左侧项目面板上的  ，从菜单中选择 Worksheet，并双击打开。

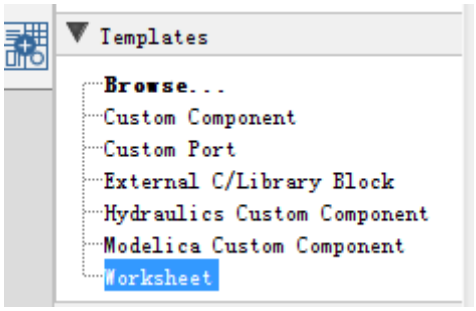


图 1-51　Templates 中的 Worksheet 模板

在 Maple 中使用如下的命令：

```
mModel := MapleSim:-LinkModel();

mModel:-GetSubstitutions();

mModel:-SetSubstitutions({`Main.R1.theta` = theta__1,

                          `Main.R2.theta` = theta__2,

                          `Main.R3.theta` = theta__3,
```

```
`Main.Crank.L` = L__1,
`Main.Crank.m` = m__1,
`Main.ConnectingRod.L` = L__2,
`Main.ConnectingRod.m` = m__2,
`Main.P1.F1` = F__1},savewithmodel=false);
MB := mModel:-GetMultibody(simplify = true)
```

注意：换行使用 Shift+Enter 键。不能直接按回车键，否则会提前执行该语句。

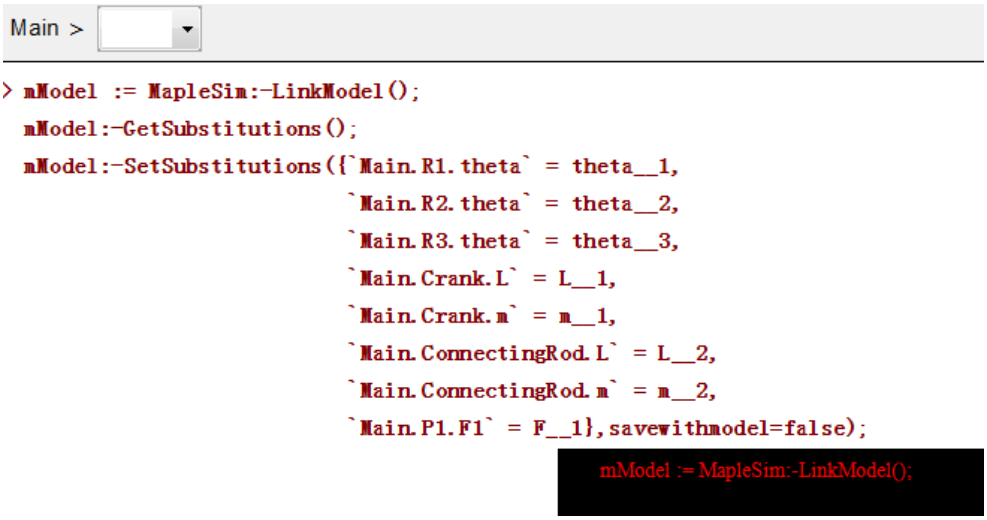


图 1-52　执行程序

更多符号分析功能，例如使用符号方程求解逆向运动学问题，另见后续模型分析，或者参考《MapleSim 系统建模和仿真》教材。

第二章 建模实践 - 四连杆机构

MapleSim 是一个多学科系统级建模仿真工具，提供图形化的设计环境，让用户在单一的环境中通过简单的拖动方式完成各种系统的建模、分析和仿真。通过 MapleSim 系统级模型，设计人员可以快速地对设计方案进行可行性分析，优化系统参数并进行故障诊断的前提设计，减少基于物理样机实验的巨大时间和费用。

本次体验活动环节通过几个实例的练习，加深了解 MapleSim 的独特之处。

2.1 实训 1. 单摆模型

模型下载地址：http://www.maplesoft.com.cn/book/2.1.zip

该示例主要体现 MapleSim 物理系统建模方式，或基于物理对象建模的特点。方便的建模方式，可以快速的搭建复杂的系统模型。

单摆模型如下所示，由固定端、转动副和连杆（质心位于中心，长度为 L）组成。

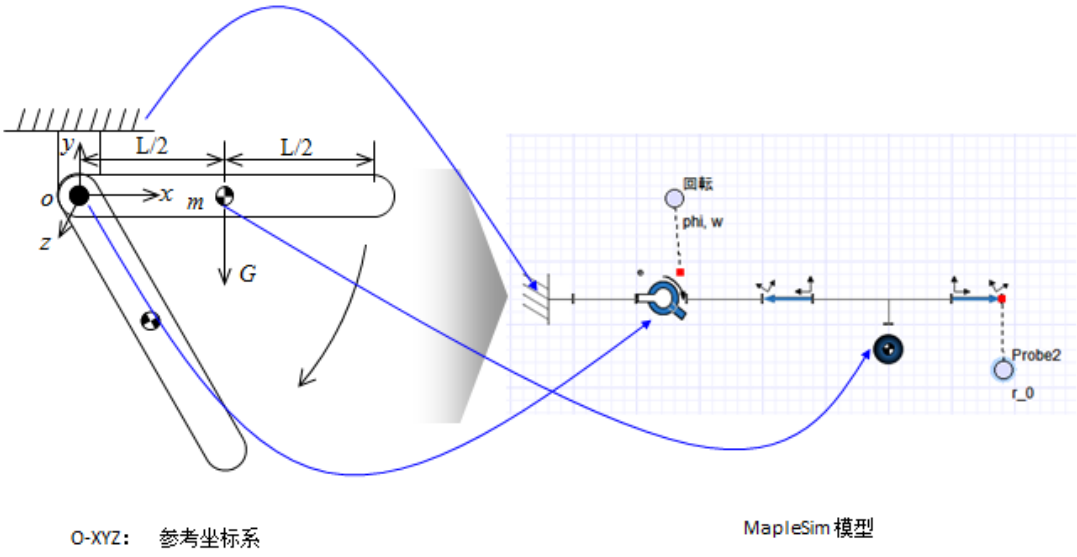


图 2-1 将物理模型转化为 Maplesim 模型

所用到的元件有：

表 2-1　模型中的元件

名称	图标	功能描述	数量
Fixed Frame		固定端	1
Revolute		转动副	1
Rigid Body Frame		刚体坐标系元件	2
Rigid Body		刚体元件质心	1

1. 添加元件到模型工作区中，如下所示；

①Fixed Frame：单摆模型固定端；

②Revolute：单摆转动副；

④Rigid Body：单摆连杆质心；

③⑤Rigid Body Frame:连接质心的连杆。

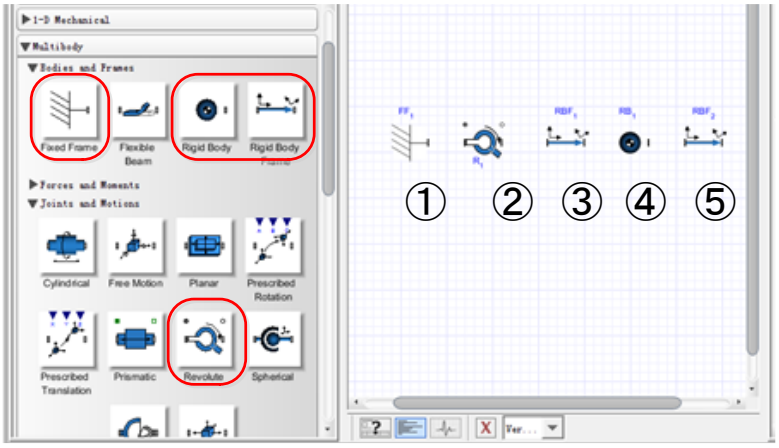


图 2-2　模型中的元件

2. 调整元件布局，如下所示：

元件调整方式：右旋转：Ctrl＋R，左旋转：Ctrl＋L，左右反转：Ctrl＋H，垂直反转 Ctrl＋F。

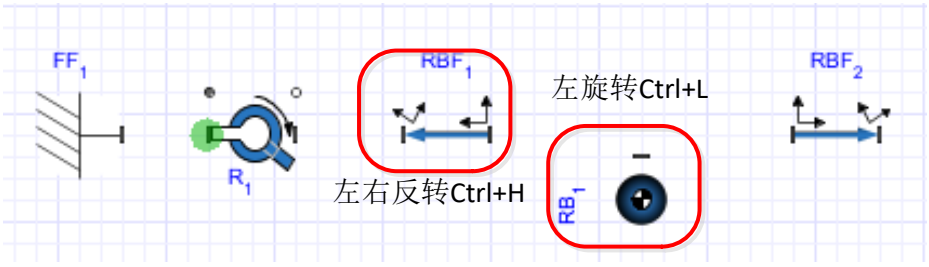


图 2-3　各个元件的位置及方向

3. 将各个元件按顺序连线，如下图所示；

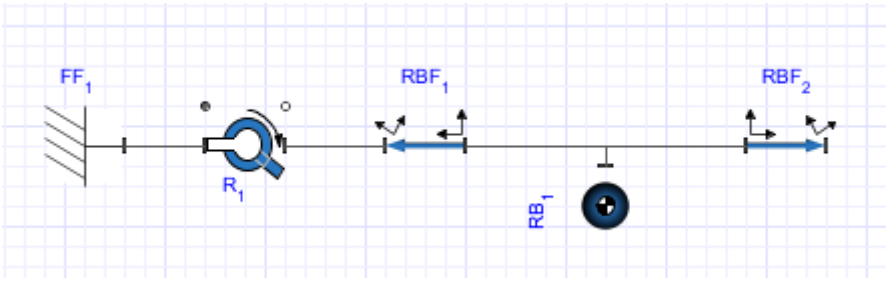


图 2-4 连线后的模型

4. 修改模型参数，如下图所示：

选中左侧的"Rigid Body Frame" 元件，从 Properties 面板上找到对应的参数项 "$\bar{r}_{XYZ}$"，将参数值[1,0,0] 修改为[-0.5,0,0]，也就是定义质心左侧的长度为 0.5m；

同样，选中右侧的"Rigid Body Frame" 元件，将参数项"$\bar{r}_{XYZ}$"的值从[1,0,0]修改为[0.5,0,0]，也就是定义质心右侧的长度为 0.5。

由此定义连杆的总长 L 为 1m，质心位于连杆的中心。

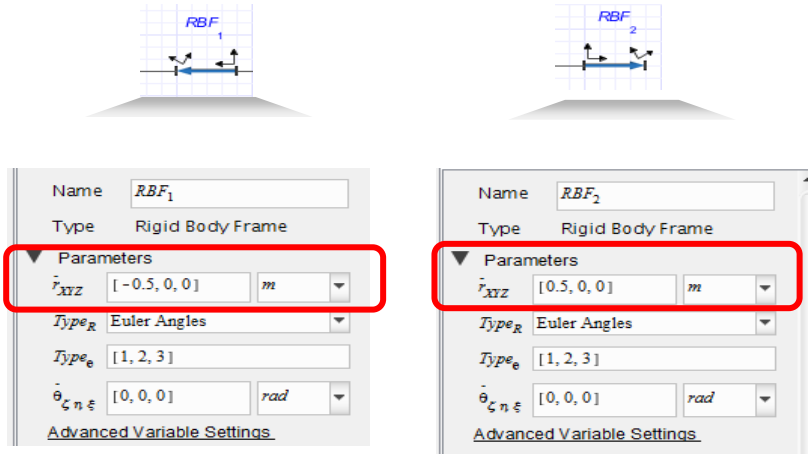


图 2-5　定义 RBF 的参数

5. 添加和设置探针 Probe，如下图所示；

点击工具栏上的“Attach Probe”图标，光标将变为探针形状；

- 点击“Revolute Joint”元件右上角的空心圆端口，添加一个探针 Probe1；
- 选择要测量的物理量：Angle 和 Speed。

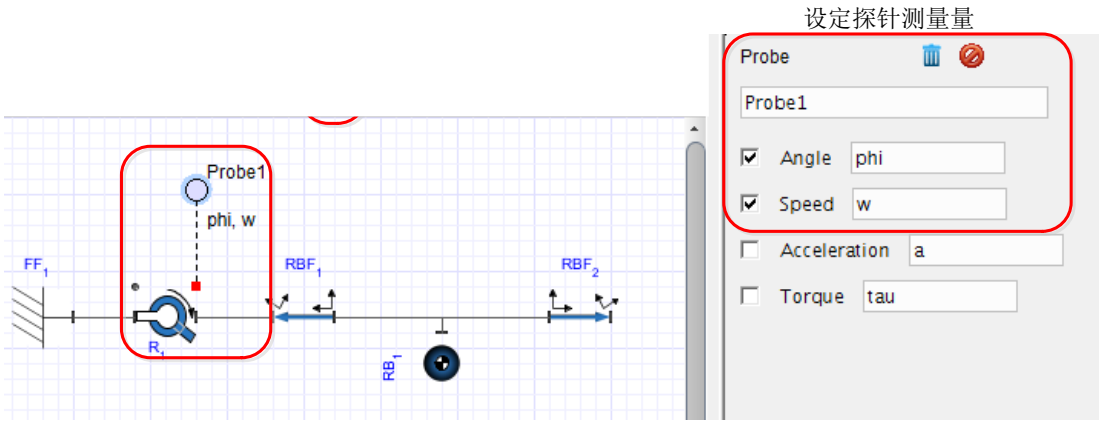


图 2-6　添加探针 Probe1

- 使用相同的方法，添加探针 Probe2 到 Rigid Body Frame（RBF_2）元件。
- 选择要输出的物理量。这里选择 Length 下的 1 和 2，也就是端点的 X 坐标和 Y 坐标。

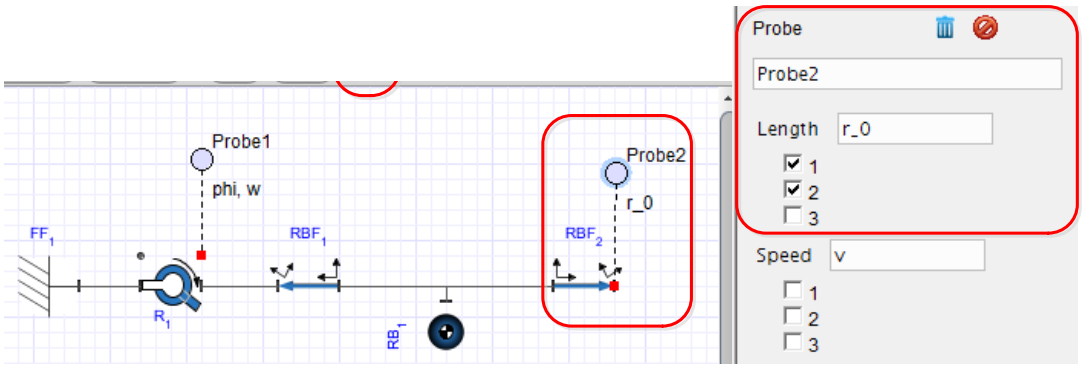


图 2-7　添加探针 Probe2

6. 运行仿真，如下图所示；

点击（▶）运行按钮，运行模型，查看仿真结果包括单摆转角角度、角速度、连杆末端 X 方向位移、Y 方向位移。

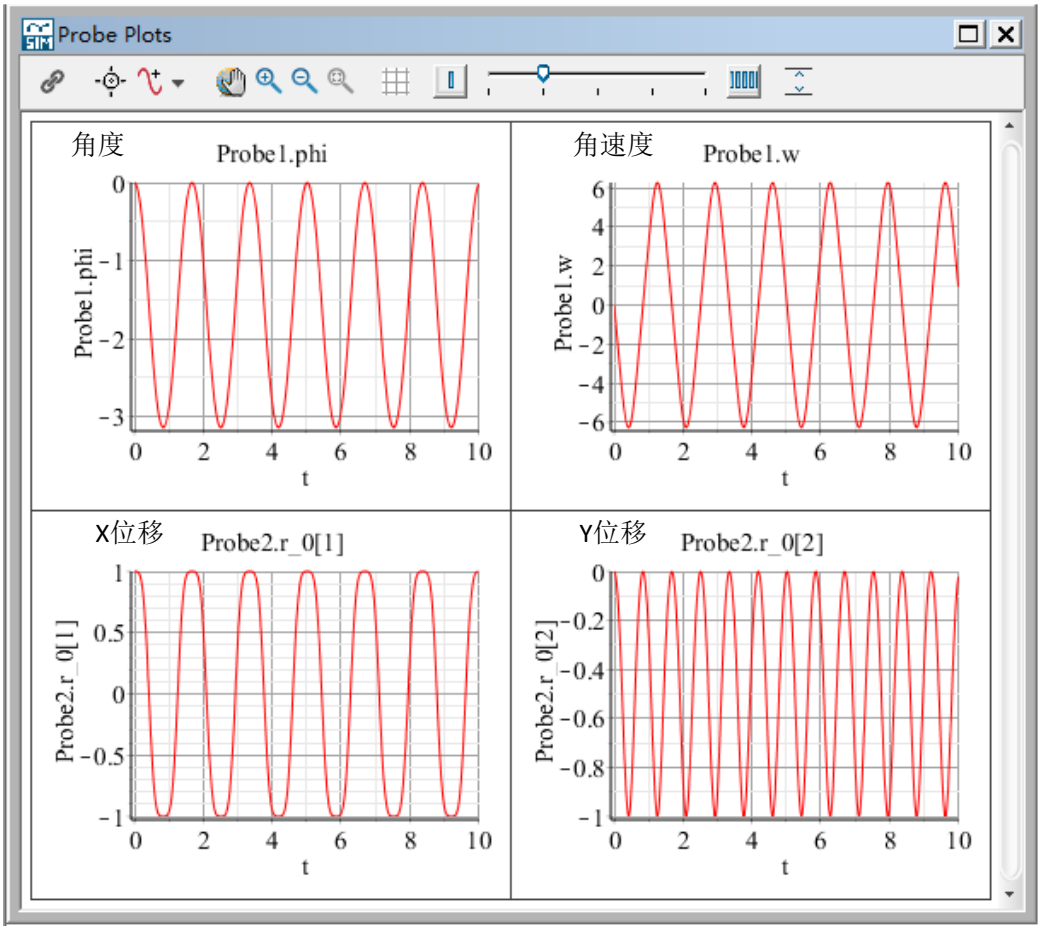


图 2-8　仿真结果图

2.2 实训 2. 考虑转动副摩擦的单摆模型

该示例主要体现 MapleSim 方便的自定义元件——可以轻松、快速地创建基于数学模型的自定义建模元件，可扩展 MapleSim 的建模元件库，定义企业专业级的元件库。

转动副摩擦产生的力矩 τ 等于摩擦系数 k 乘以转角速度 $\dot{\theta}$，如下式所示:

$$\tau(t)=k\left(\frac{d}{dt}\theta(t)\right)$$

基于转动副摩擦受力方程，采用基于方程的自定义元件，创建一个转动副摩擦受力元件。

1. 调用 Maple 自定义元件分析模板

➢ 点击左侧项目面板上的“Add Apps and Templates”图标（  ），调用 Maple 的分析

模板；

- 选择自定义元件模板：Custom Component，并双击点击，之后点击 ，建立 MapleSim 和 Maple 的连接，自动打开 Maple 自定义元件模板。

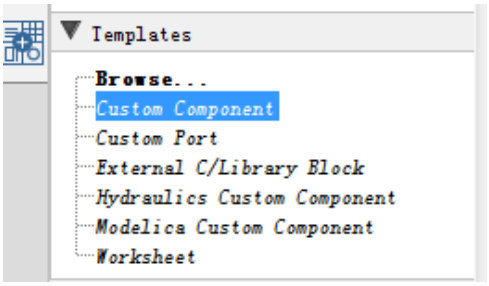


图 2-9　Custom Component 模板

2. 输入自定义模型方程；

Maple 自定义元件模板 DAE Custom Component Template 如下所示；

- 将 Equations 下的方程 eq 输入如下（eq:=[tau(t)=k*diff(theta(t),t)]），并按回车执行该方程，如下图所示。

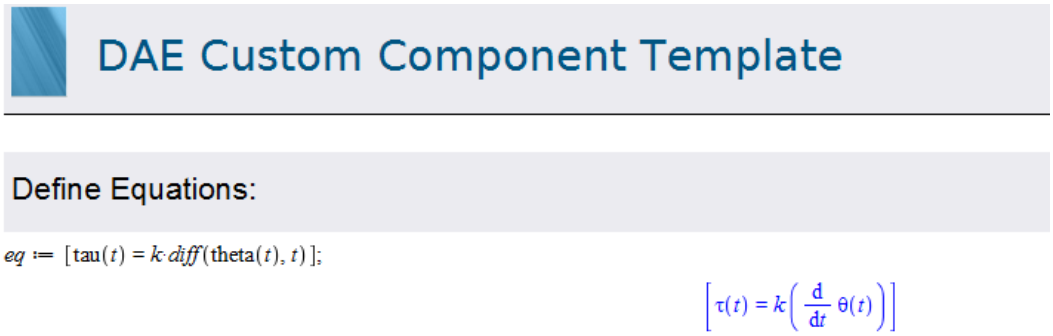


图 2-10　输入函数方程

- 点击 Configuration 下的“全部刷新”识别方程中的变量和参数，并点击 parameters 查看参数，含有时间 t 的为变量，如 tau(t)、theta(t)；不含时间 t 的为参数，如 k。如下图所示。

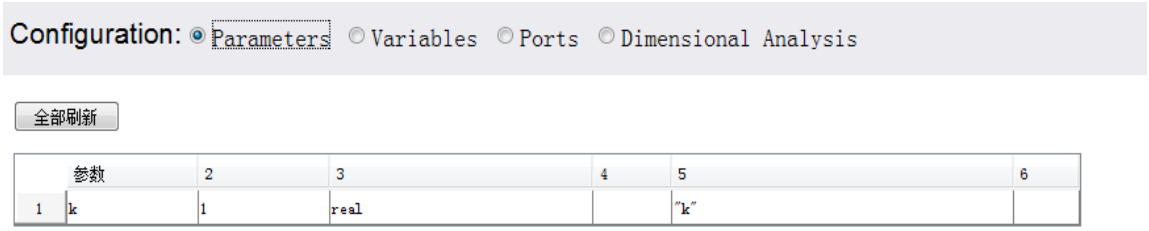


图 2-11 方程中的参数

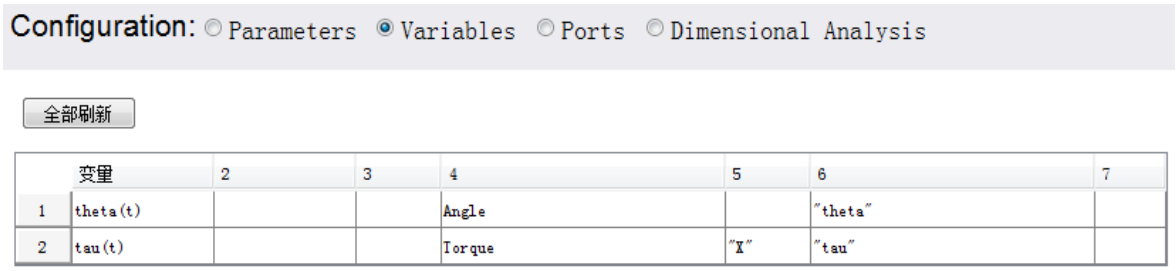


图 2-12 方程中的变量

3. 指定方程的关系，及指定变量的类型；

➢ 选择 Ports，指定端口变量。将方程中的各个变量指定到自定义元件的端口。

在该方程中，转角速度$\dot{\theta}$和转角力矩τ，作用在同一个端口，可用一个转动类型端口定义，因此该子定义元件只保留一个端口。

选择要删除的端口 pin_p,然后选择 Delete Port 删除该端口，同理选择端口 pin_n,然后选择 Delete Port 删除该端口。选择自定义元件的图标为默认（Use default）。

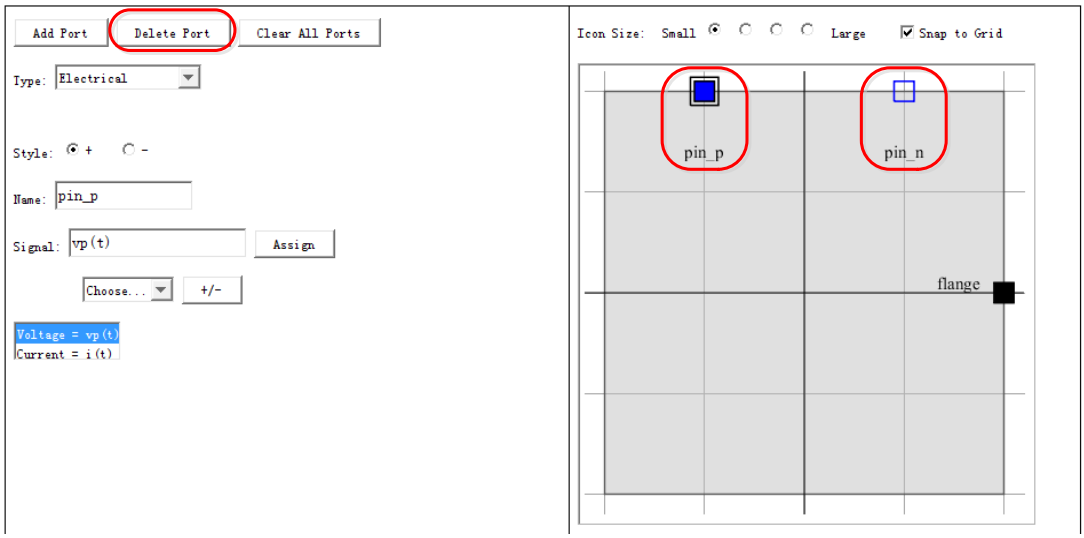


图 2-13 删除多余端口

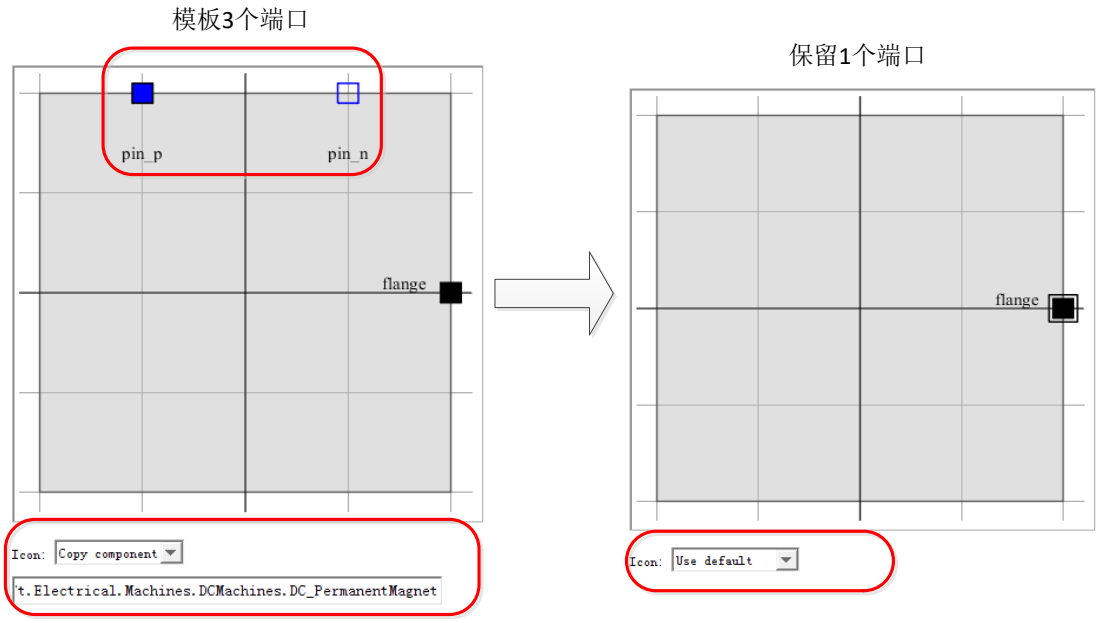


图 2-14　保留 1 个端口

定义端口 flange 变量如下（按默认）：

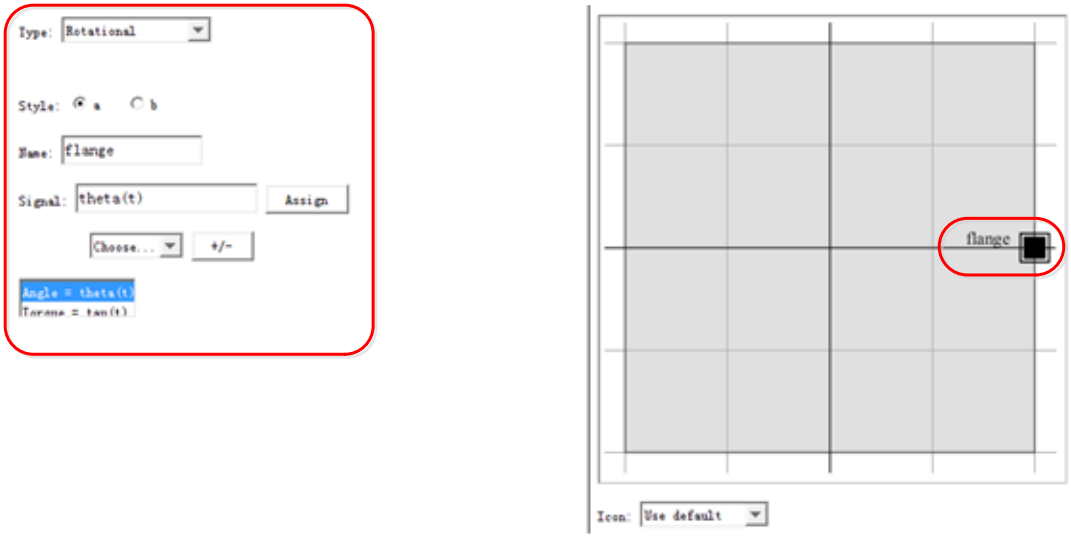


图 2-15　定义端口的类型与变量

4. 生成基于方程的自定义元件；

➢ 指定自定义元件名称。在 Component Generation 指定自定义元件的名称为 firction，并点击 Generate MapleSim Component 生成基于方程的 MapleSim 自定义元件。生成的自定义元件在 MapleSim 的 Local Component> Components>friction，如下图所示。

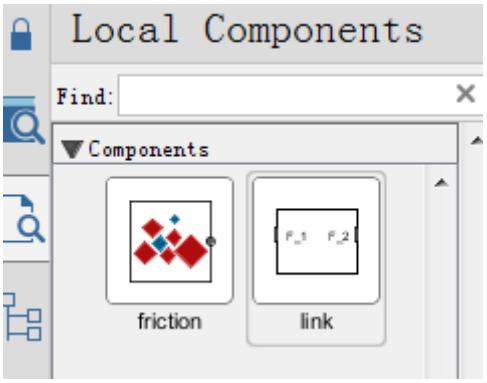


图 2-16　自定义元件

5. 考虑摩擦的单摆模型；

➢ 将 friction 元件添加到模型中（默认摩擦系数 k 为 1，可自行修改），并按如下连线。

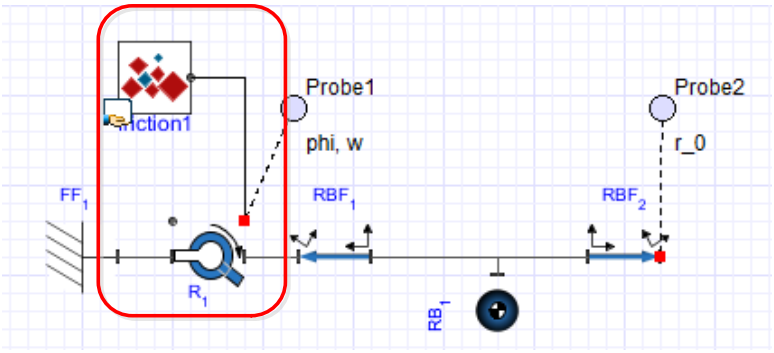


图 2-17　添加自定义元件

6. 运行仿真，查看带摩擦后的运动效果；

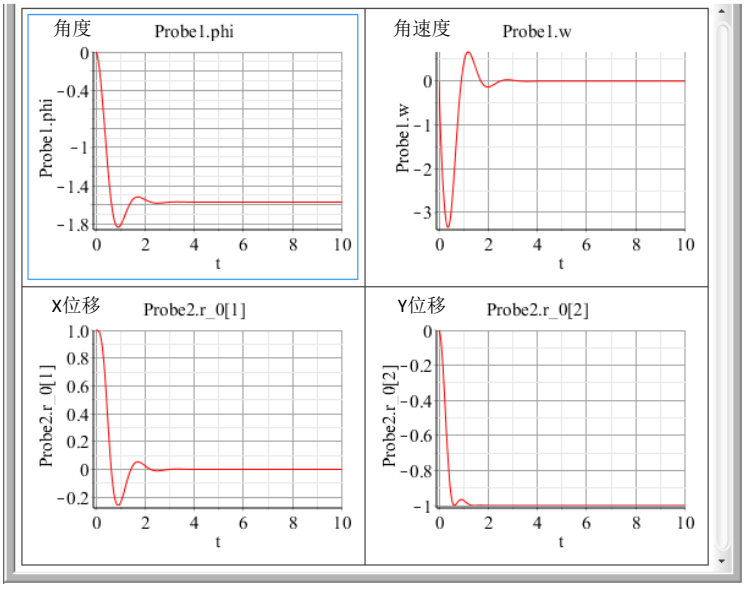


图 2-18　仿真结果图

2.3 实训 3. 平面四连杆模型

该示例主要体现 MapleSim 物理系统建模方式，在扩展到复杂系统时的便利性和可继承性。

四连杆机构四个连杆 L1，L2，L3，L4 和四个转动副 R1，R2，R3，R4 组成。如下图所示。

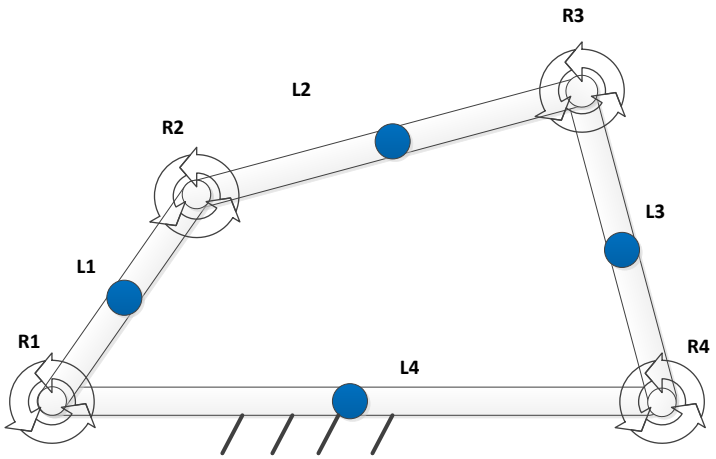


图 2-19　四杆机构的物理模型

1. 创建连杆子系统；

四杆机构的各个连杆结构一样，只是长度不同，因此可以将已有的连杆建立成子系统，并扩展到四个连杆。

➢ 删除 Probe2 探针,并框选连杆元件，如下图所示。

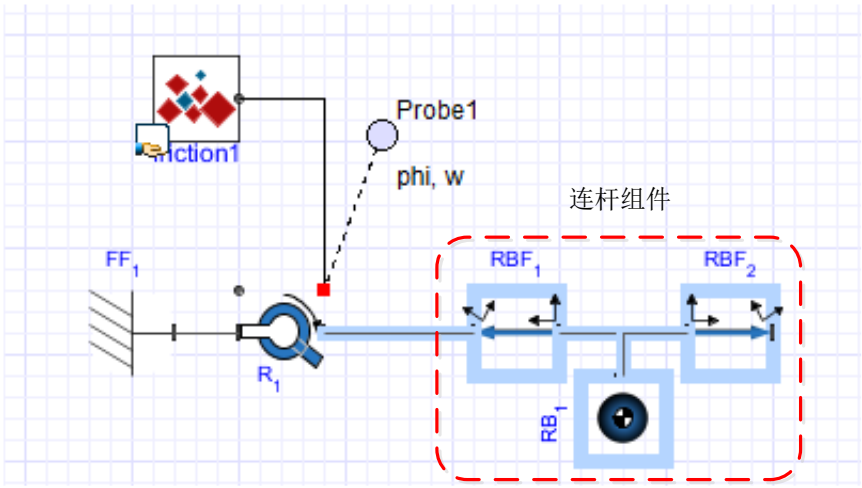


图 2-20　创建连杆子系统

➢ Ctrl+G 将连杆组件封装到子系统中，并指定子系统的名称为 link，如下图所示。

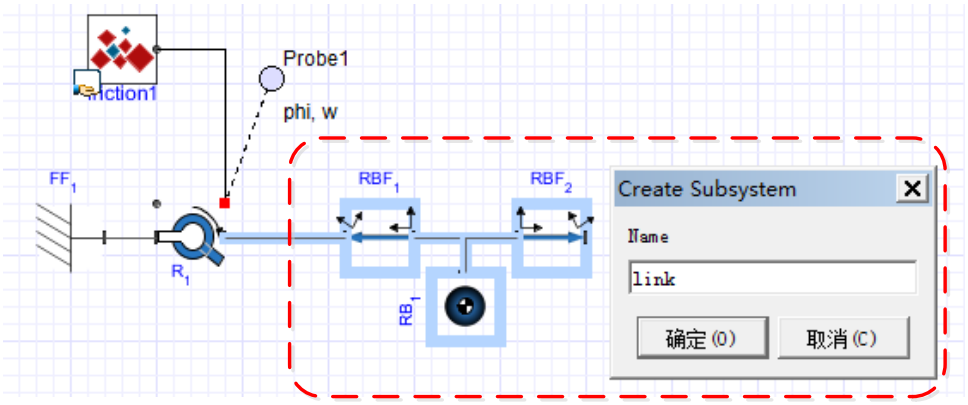


图 2-21　输入子系统名称

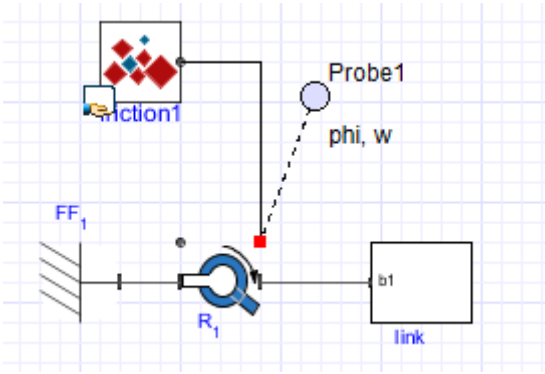


图 2-22　创建子系统后的模型

➢ 双击 link 模块，进入 link 子系统中，将连杆的右侧端口连接到 link 子系统的端口上，如下图所示。

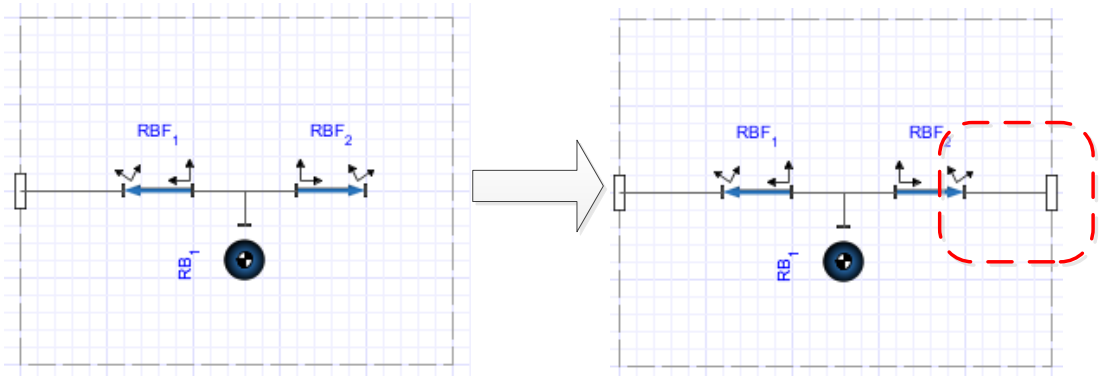


图 2-23　创建子系统的端口

2. 定义 link 子系统的参数；

➢ 用符号变量 L 来定义连杆的长度，按质心位于中心的布置，则将质心左侧的连杆

长度定义为-L/2,质心右侧的连杆长度定义为 L/2，如下图所示。

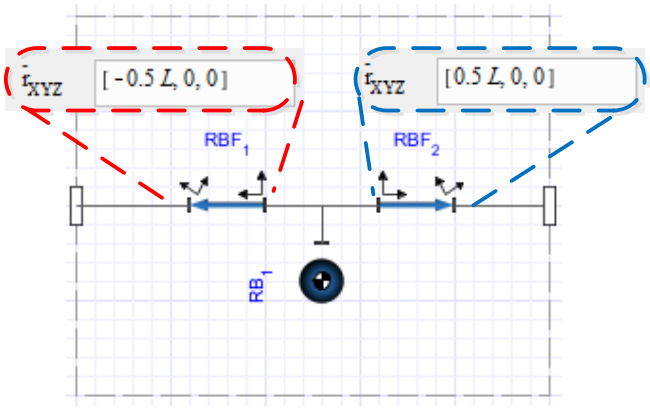


图 2-24　定义 RBF 的参数

- 定义 link 子系统的连杆参数。在 link 子系统下点击参数界面 Parameters，进入参数设置界面，并在子系统参数名称中输入 L，用于指定 L 的数值。

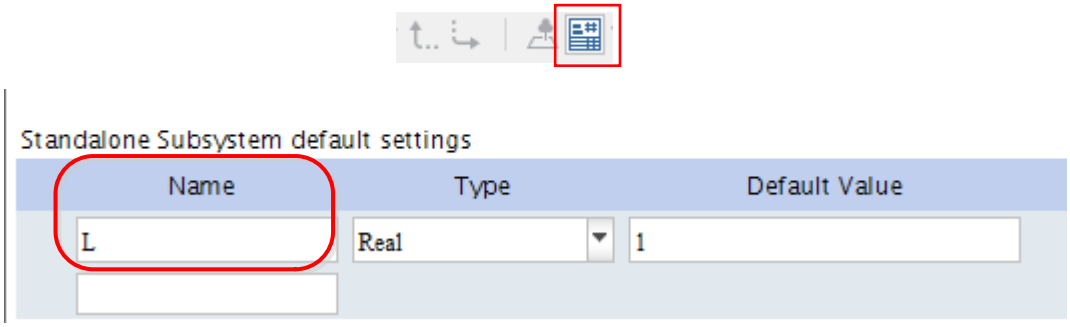


图 2-25　定义子系统的参数

- 查看 link 子系统参数。在 link 子系统下点击模型界面 Diagram,回到建模界面上，点击 link 子系统在右侧的 Inspector 下查看子系统的参数 L 数值默认为 1（可修改），如下图所示。

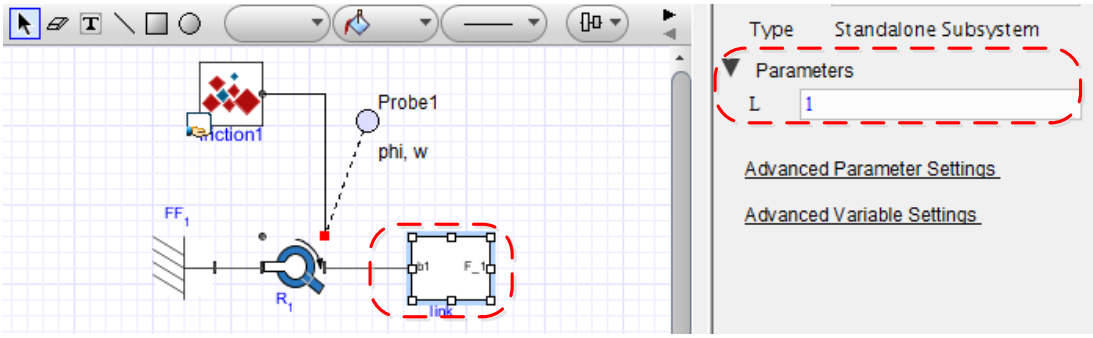


图 2-26　子系统的参数显示

3. 共享 link 子系统；

将连杆复制到多个，同时能指定不同连杆的不一样的长度，需要将 link 复制为共享子系统。

- Ctrl+C 复制 link 子系统，Ctrl+V 粘贴 link 子系统，并在弹出的 link 子系统属性中选择为共享子系统（默认）。

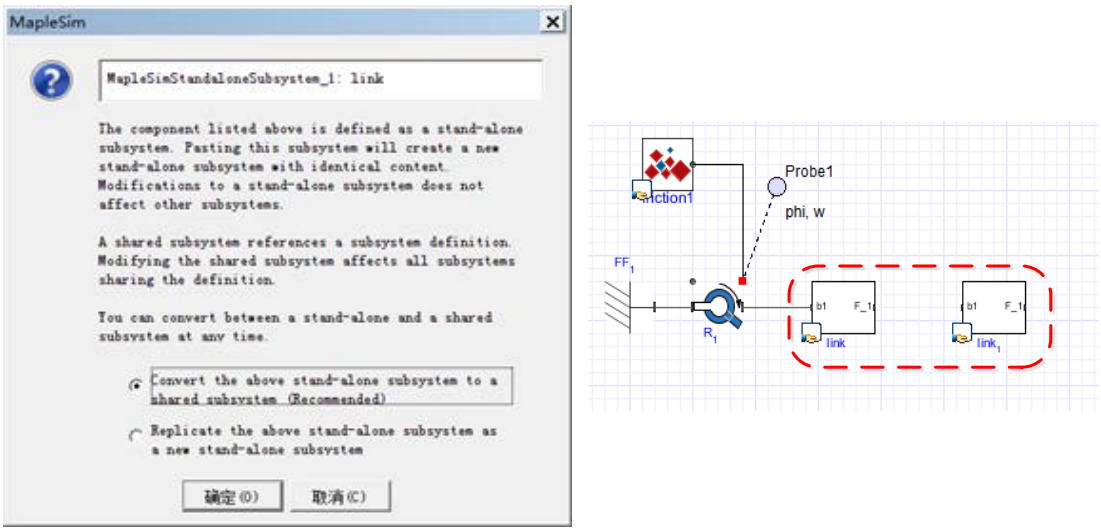


图 2-27　变为共享子系统

- 将共享的 link 子系统复制到 4 个，如下图所示。

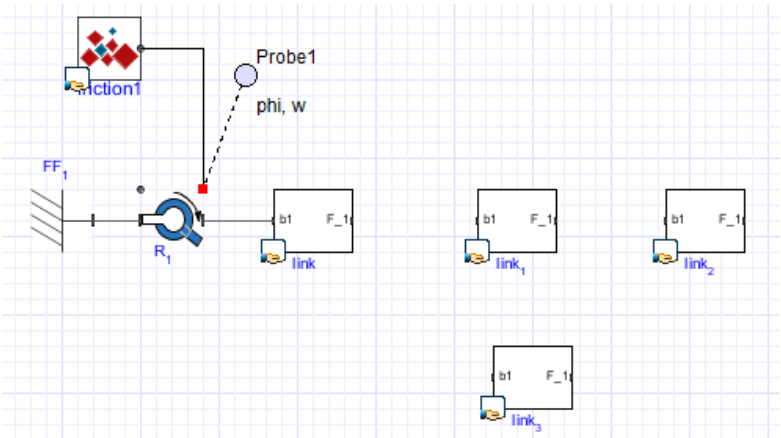


图 2-28　复制多个子系统

4. 添加剩余的转动副，并连接；

- 在工作区中添加三个转动副（Revolute），并按如下布置

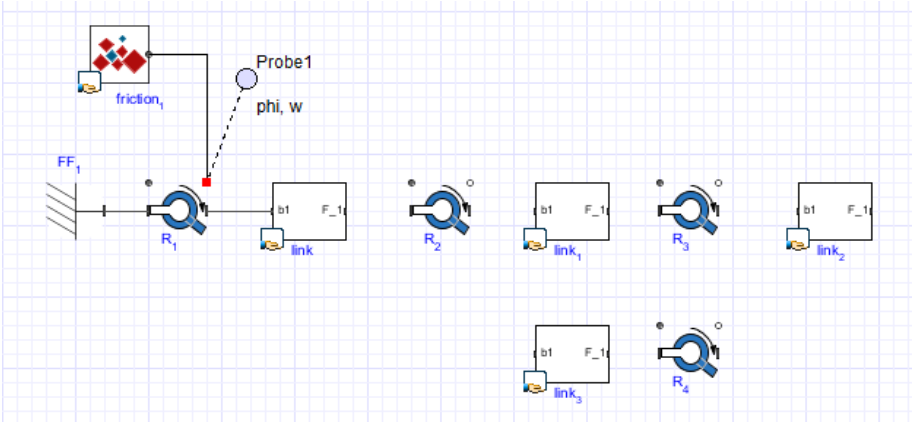


图 2-29　各个元件位置

➢ 将转动副和连杆按顺序连接，构成封闭的四连杆机构，如下图所示。

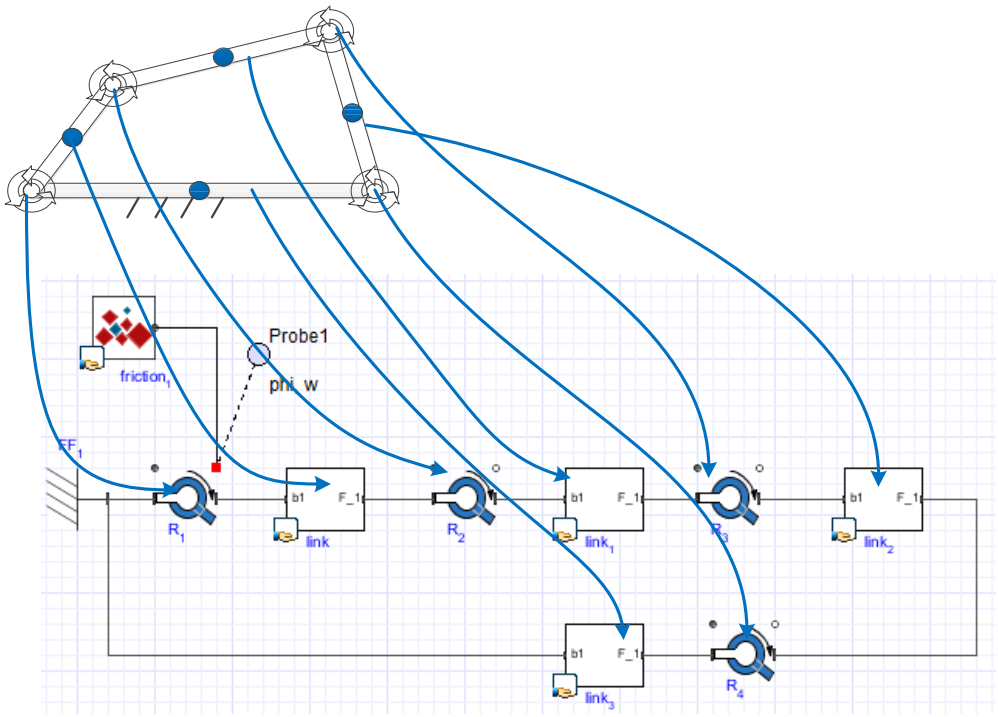


图 2-30　模型各元件与物理模型相对应

5. 修改各连杆长度；

➢ 修改各连杆的长度，如 L_1=1，L_2=2，L_3=3，L_4=3;分别点击连杆子系统 link,并在右边的参数修改界面下修改 L 数值，如下图所示。

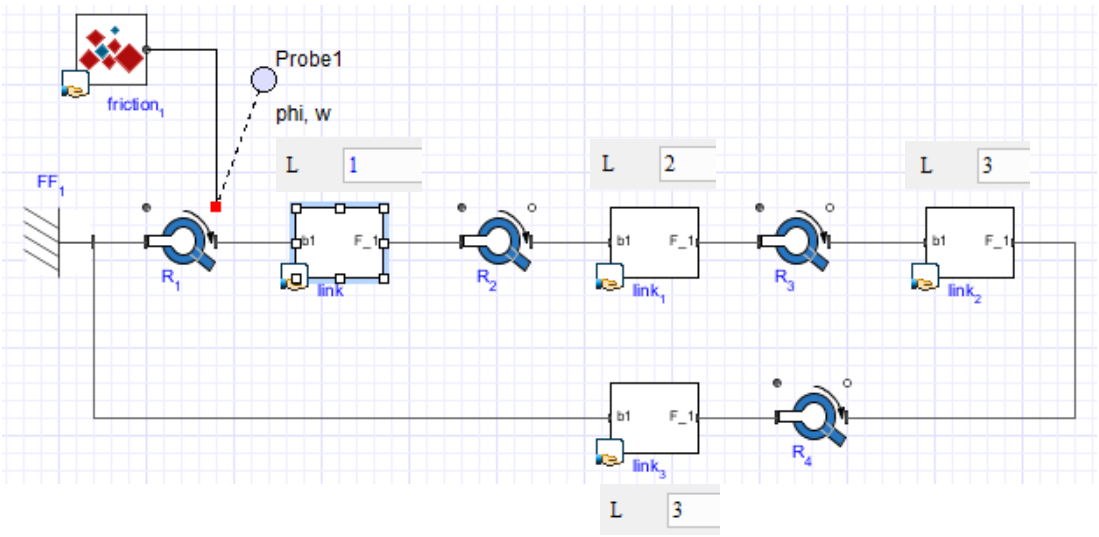


图 2-31　定义各个子系统的参数

6. 设定连杆初始位置，并运行模型；

- 设定第一连杆初始角度为 60 度：选择第一转动副 R1，初始角 θ_0 定义为 60 度，初始状态类型 $IC_{\theta,\omega}$ 设为 Treat as Guess，即四连杆机构的初始位置在 60 度附件寻找最优解（另有 Ignore 忽略初始状态值，Strictly Enforce 强制初始状态值）。完成初始值设定后运行模型，如下图所示。

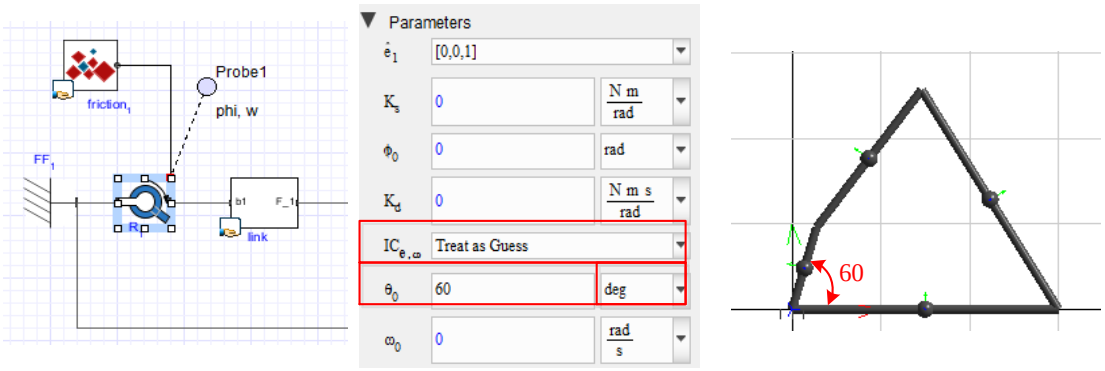


图 2-32　定义模型初始条件

2.4 实训 4. 平面四连杆电机控制

该示例主要体现 MapleSim 多学科系统建模仿真的特性，该示例集成了机械电子、信号和控制为一体，实现机电一体化控制。

打开 fourbar_control.msim 文件，如下图所示，四连杆机构的运动由直流永磁电机驱动，电机输出的力矩大小由输入的电压决定。修改电机电压输入值 constant，可查看不同的电压下机构的运行情况。

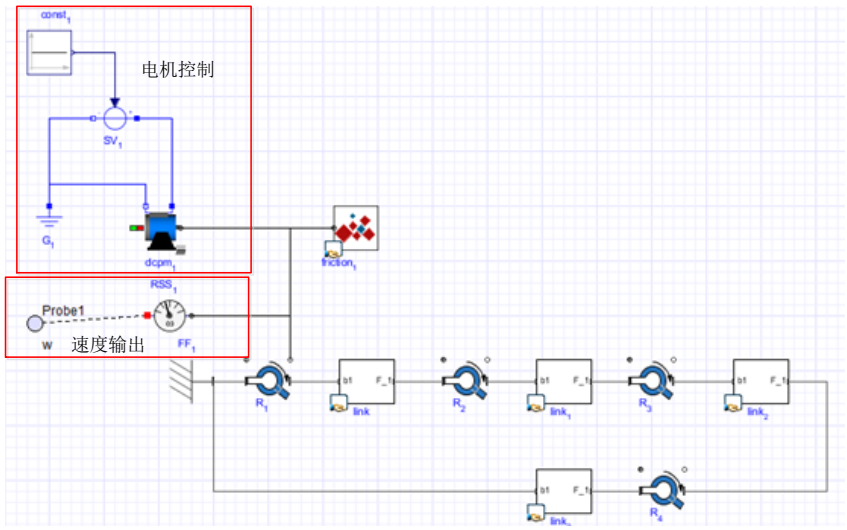


图 2-33 添加电机驱动模型

打开 fourbar_PIDcontrol.msim 文件，如下图所示，采用 PID 算法控制四连杆机构。

以四连杆机构及电机模型为整体作为系统的被控对象，封装到 fourbar 子系统内，系统的输入为电机电压输入，系统的输出为第一连杆角速度输出，按 PID 算法原理模型将期望信号（Step）、反馈接口（Feedback）、PID 元器件和被控对象连接如下。

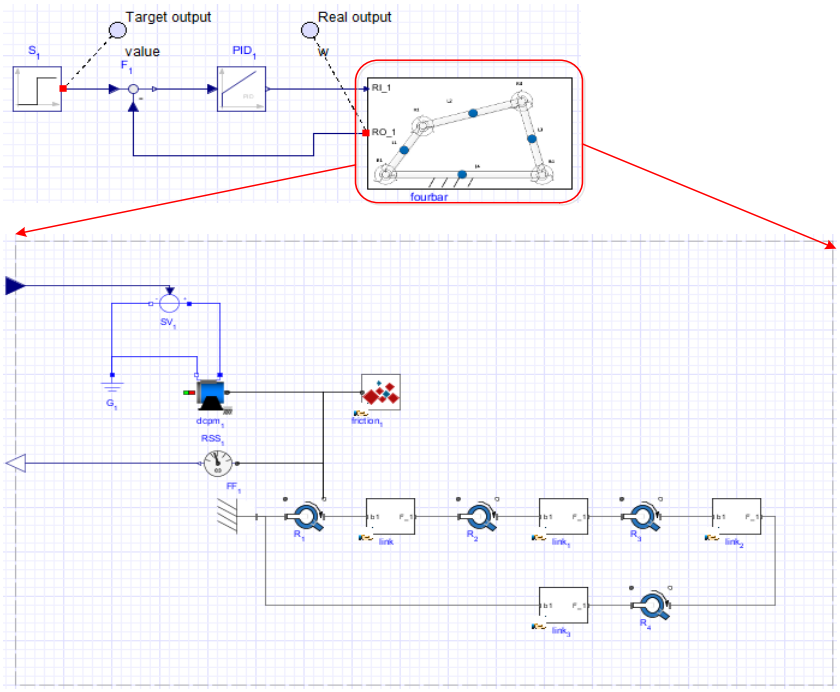


图 2-34 创建 fourbar 子系统

PID 控制中，可针对不同的期望值，调节 PID 中的参数，以获得良好的系统的控制效果。如下图所示：

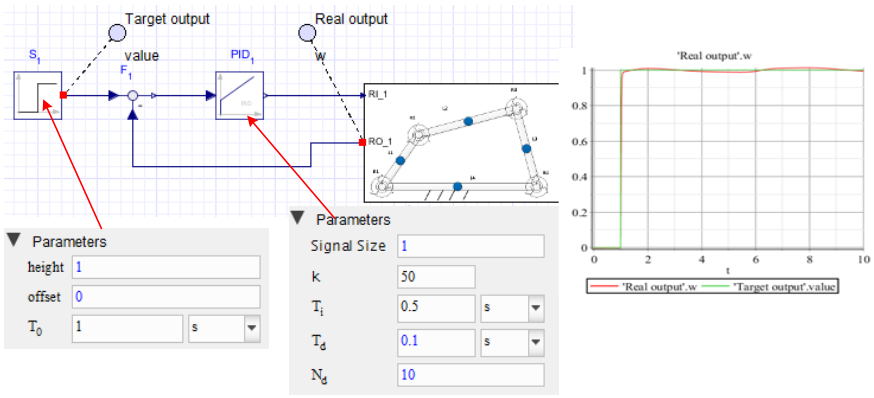


图 2-35　定义模型中各元件参数

2.5 实训 5. 平面四连杆电机模型输出

该示例主要体现利用 Maple 对 MapleSim 模型进行高级分析，实现 MapleSim“白盒子”模型的特点，即可输出模型的系统方程及输出高效的模型代码。系统方程用于理论验证，运动控制等，模型代码输出包含 C，C++，Simulink、Labview，FMI，DSpace 等。

1. 系统方程输出

打开 fourbar_connector.msim 文件，分析四连杆控制模型的方程（fourbar 子系统），系统方程输出需要用到 Maple 的 Equations 分析模板实现。

- 点击左侧项目面板上的“Add Apps and Templates”图标，调用 Maple 的分析模板；
- 选择自定义元件模板：Equations Extraction，并在双击，建立 maplesim 和 maple 的连接，自动打开 Maple 方程提取模板。

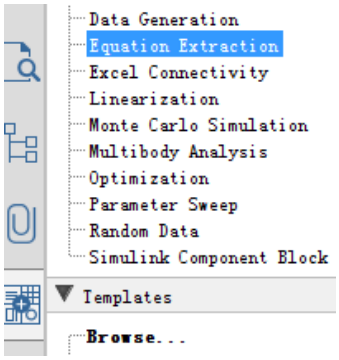


图 2-36　Equations Extraction 模板

- 选择被控对象 fourbar 子系统：在 Equation Display 节下的对象选择中通过向下三角箭头选择 fourbar 子系统，并点击 Load Select Subsystem 加载所选择的对象，输出对象的方程，如下图所示。

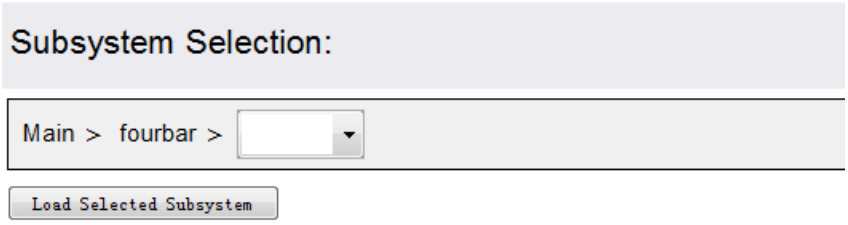


图 2-37　加载子系统

- fourbar 子系统的方程输出：在加载所选对象后，自动获得对象的方程组成，如系统变量、系统参数和系统方程，如下图所示。

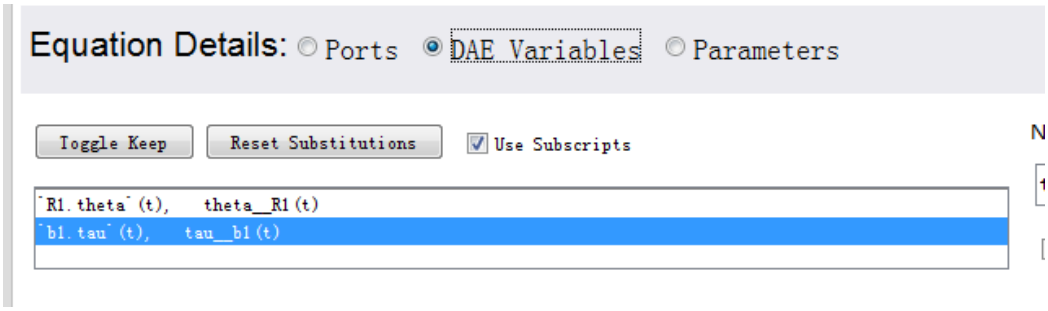
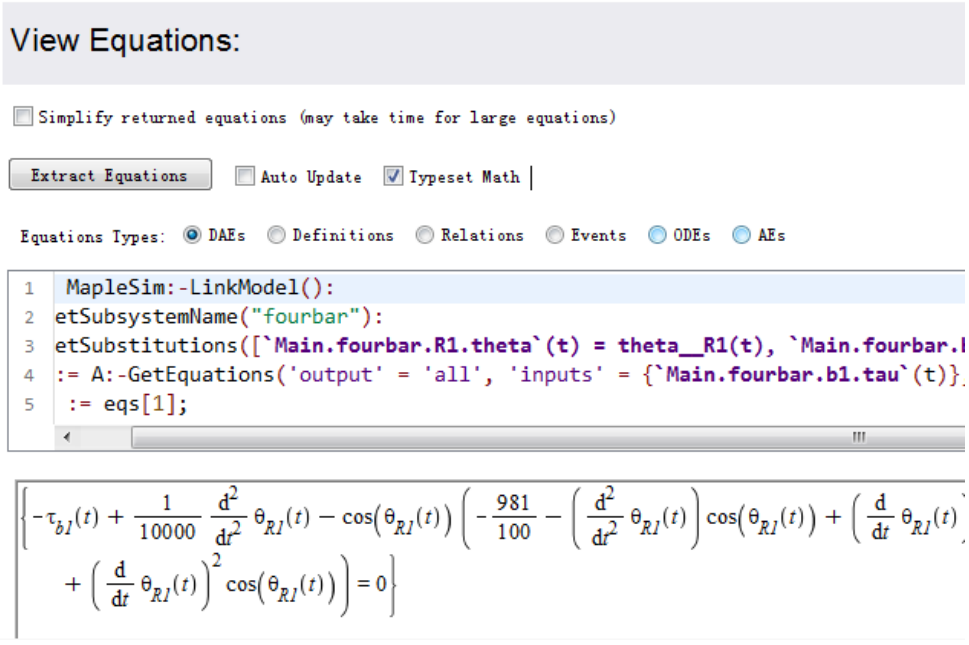


图 2-38　系统中的变量



$$\left\{-\tau_{b1}(t)+\frac{1}{10000}\frac{d^2}{dt^2}\theta_{R1}(t)-\cos\left(\theta_{R1}(t)\right)\left(-\frac{981}{100}-\left(\frac{d^2}{dt^2}\theta_{R1}(t)\right)\cos\left(\theta_{R1}(t)\right)+\left(\frac{d}{dt}\theta_{R1}(t)\right)\right.\right.$$
$$\left.\left.+\left(\frac{d}{dt}\theta_{R1}(t)\right)^2\cos\left(\theta_{R1}(t)\right)\right)=0\right\}$$

图 2-39　系统中的方程

2. C 代码生成

MapleSim 中的代码生成能够输出任意类型的模型，需要注意的是输出模型的端口必须是信号输入（RealInput）和信号输出（RealOutput）。使用 maplesim 附属的 Code Generation 模板生成的 C 代码可以再 Maple 中使用，也可以修改后集成到任意的目标中，脱离 Maple/MapleSim 独立运行。

- 点击左侧项目面板上的“Add Apps and Templates”图标，调用 Maple 的分析模板；
- 选择自定义元件模板：Code Generation，并双击，建立 MapleSim 和 Maple 的连接，自动打开 Code Generation Template 模板。

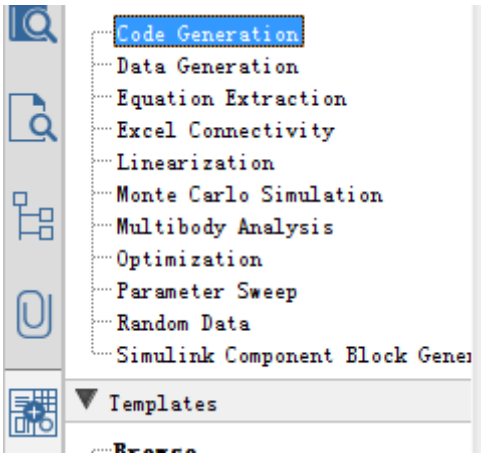


图 2-40　Code Generation Template 模板

- 选择被控对象 fourbar 子系统：在 C Code Generation 节下的对象选择中，通过向下三角箭头选择 fourbar 子系统，并点击 Load Select Subsystem 加载所选择的对象，如下图所示。

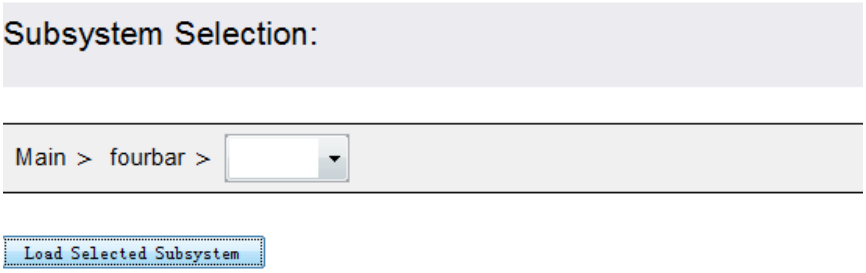


图 2-41　加载子系统

- 被控对象 fourbar 子系统的接口获取：在加载所选对象后，自动获得对象代码的交互接口，如输入接口、输出接口和系统参数，如下图所示。

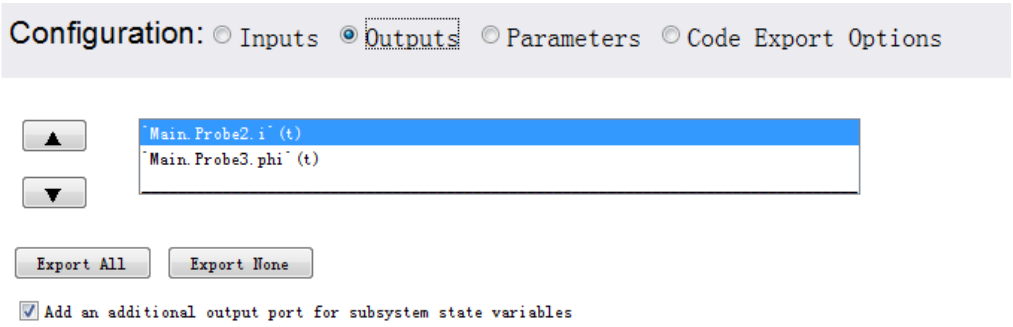


图 2-42　输出端口

- 选择 Euler 求解器，如下图所示。

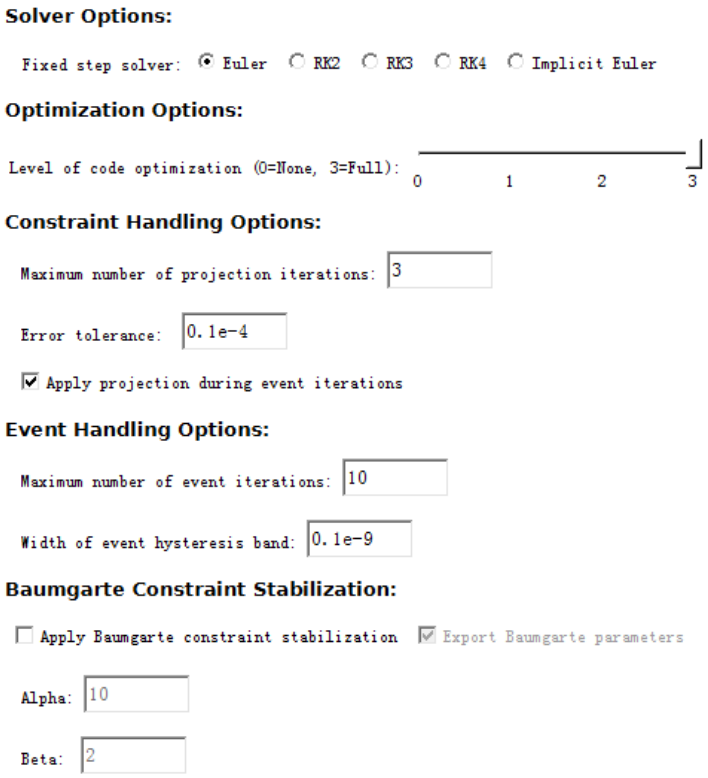


图 2-43　定义参数

- 生成并保存 C 代码，选择保存路径和文件名，并单击下面的 Gerneration C code 按钮，如下图所示。

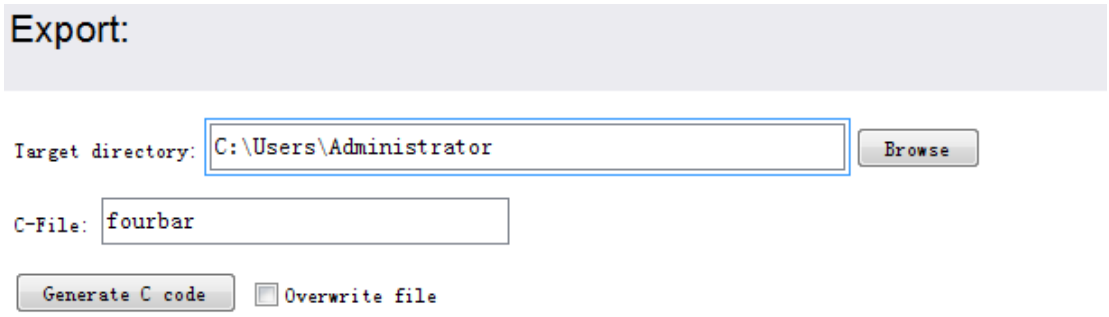


图 2-44　修改文件名称和保存位置

- C 代码将显示在 View C Code 区域，如下图所示。

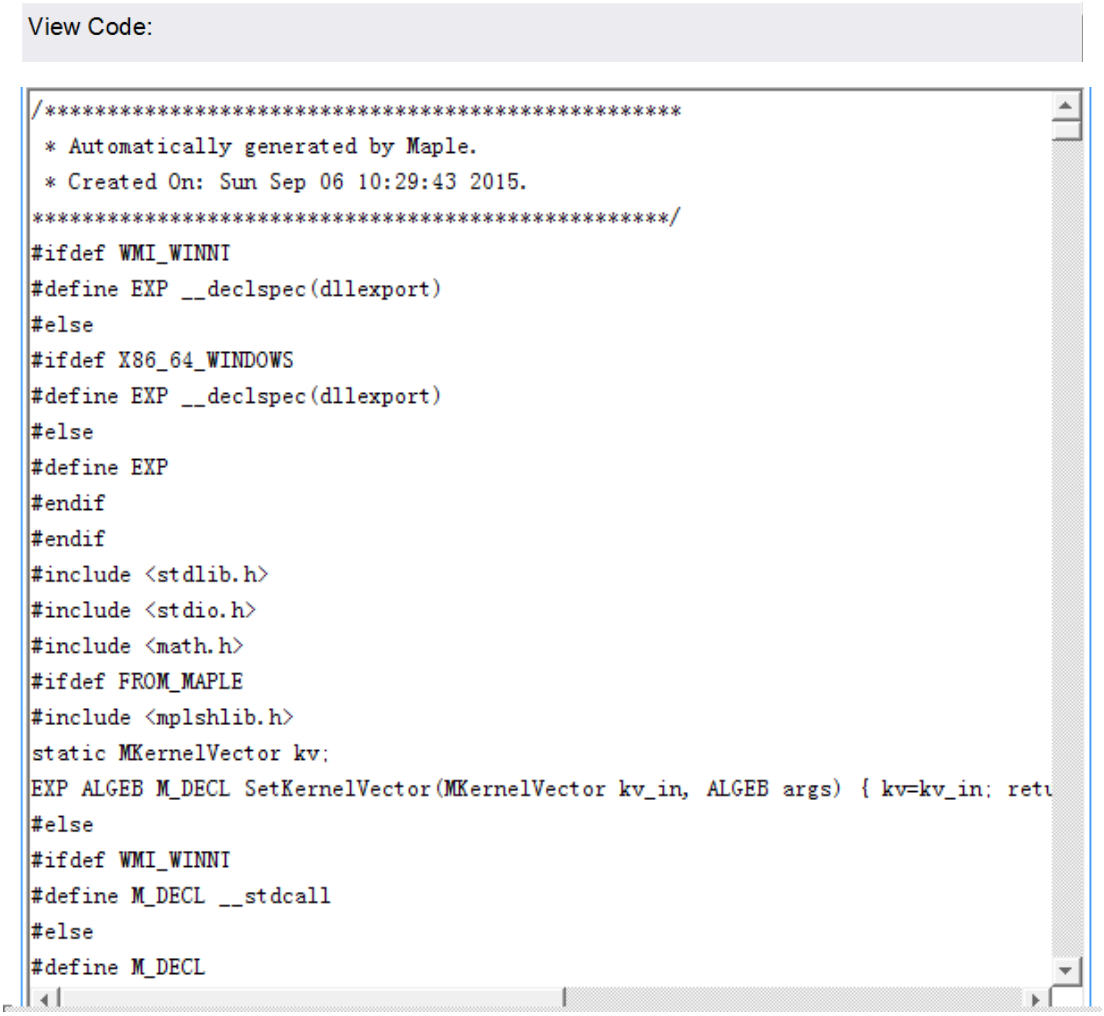


图 2-45　子系统的 C 代码

- 同时也会保存到指定的路径，如下图所示。

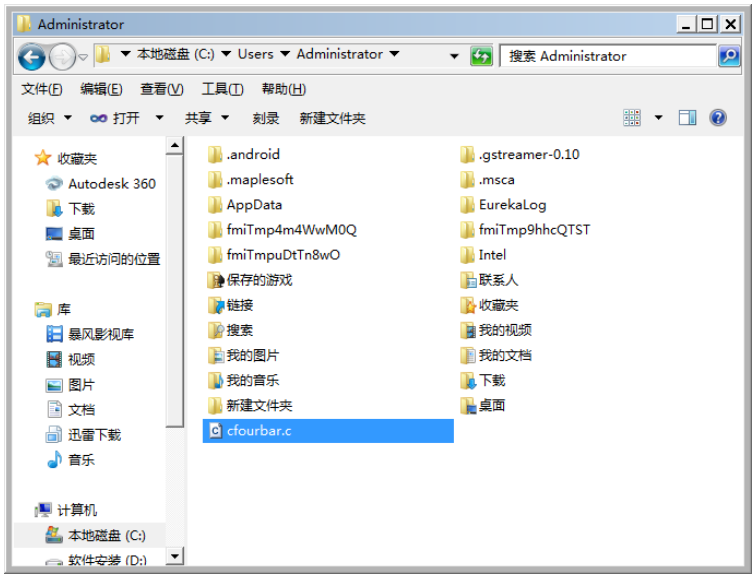


图 2-46　保存 C 代码文件

2.6 附：PID 参数整定

1. 将四连杆机构模型线性化

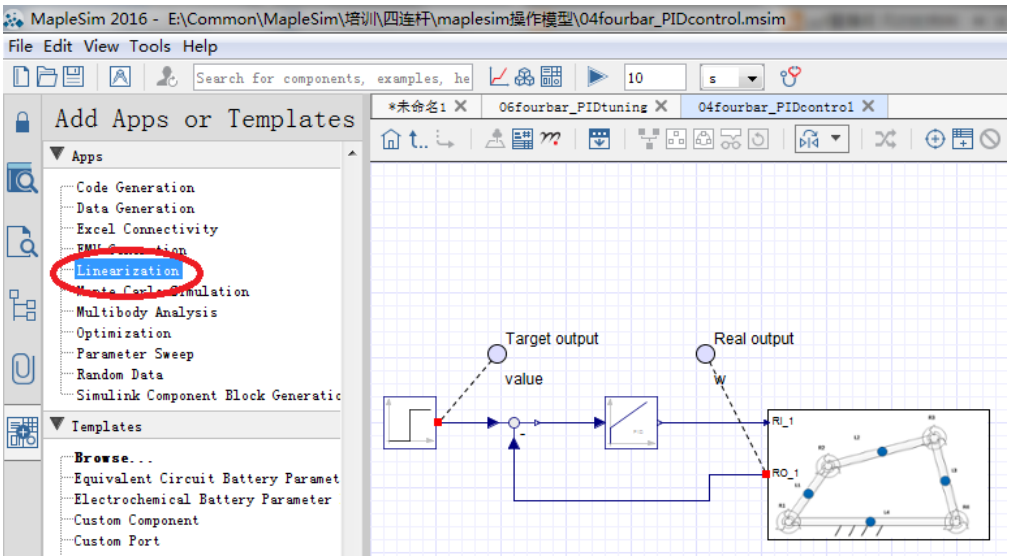


图 2-47　打开 Linearization 模板

选择 fourbar 子系统，加载子系统后，点击 Linearize。

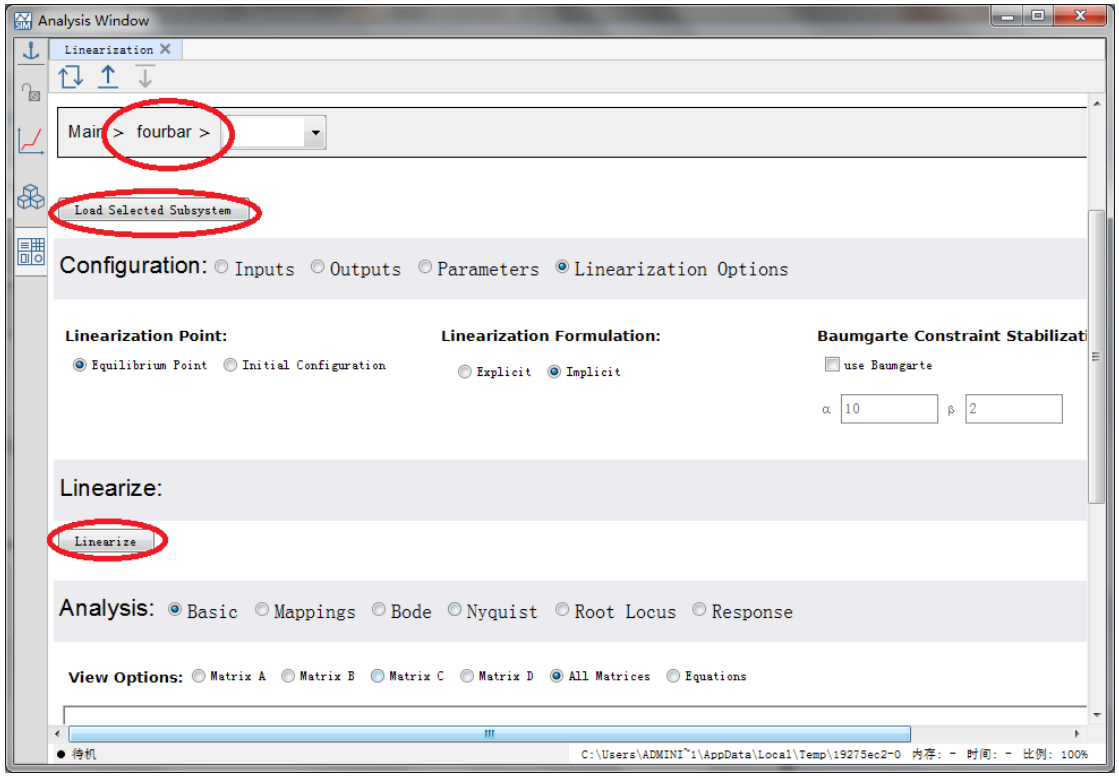


图 2-48　将子系统线性化

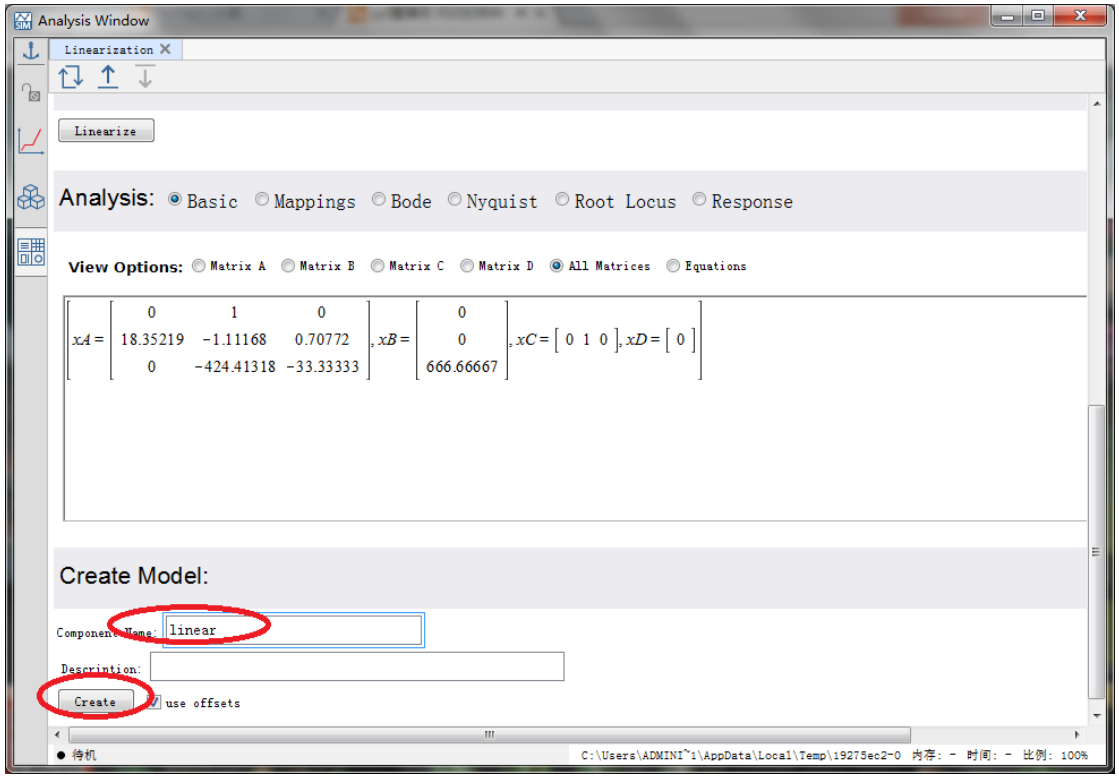


图 2-49　修改元件名称并创建

2. PID 整定

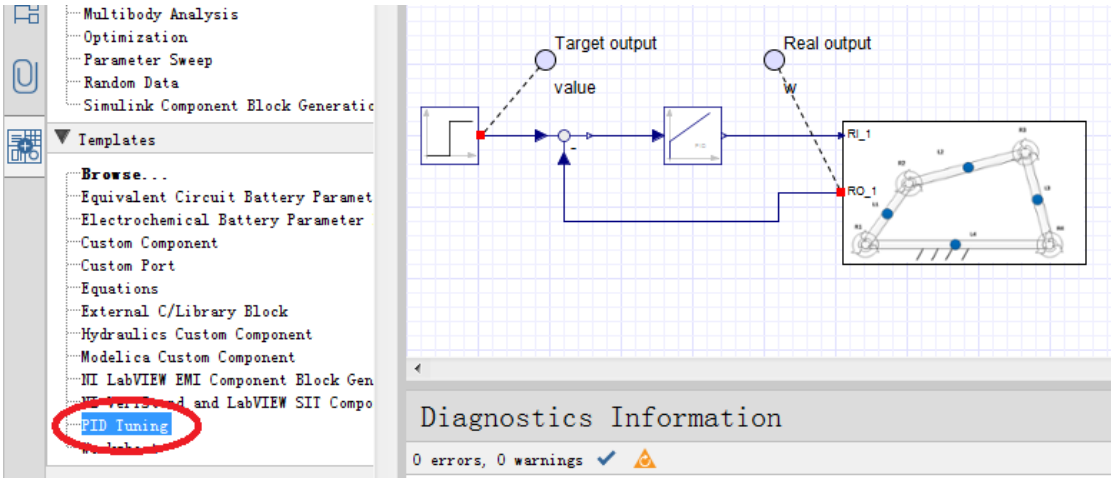


图 2-50　打开 PID Tuning 模块

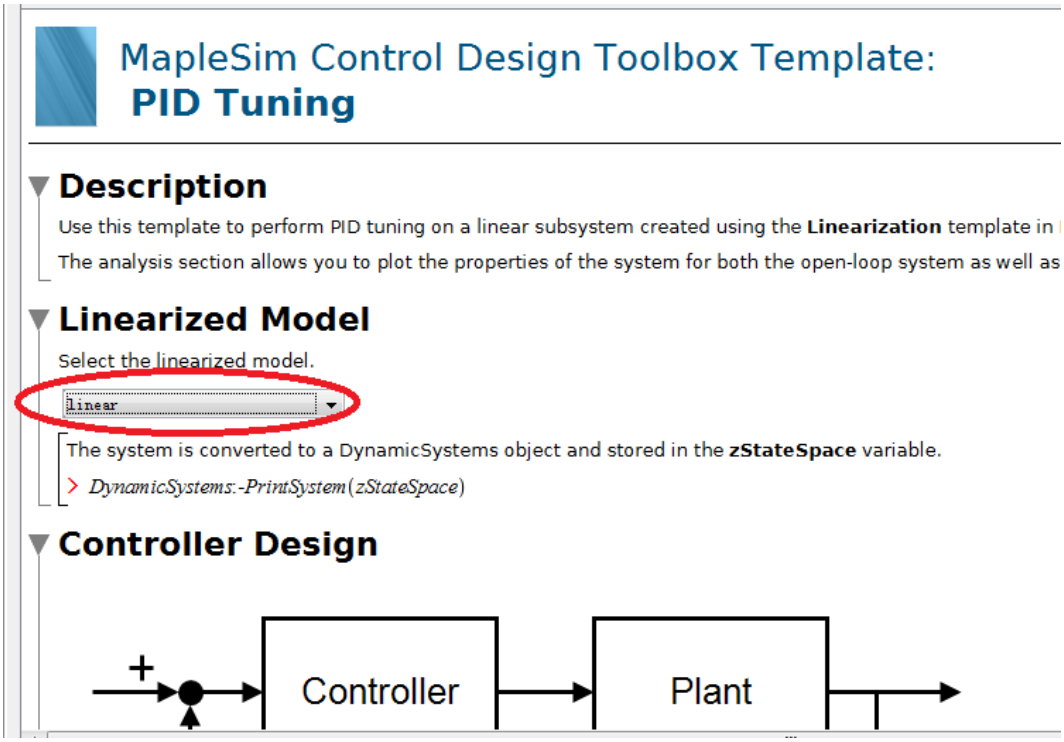


图 2-51　选择线性化元件

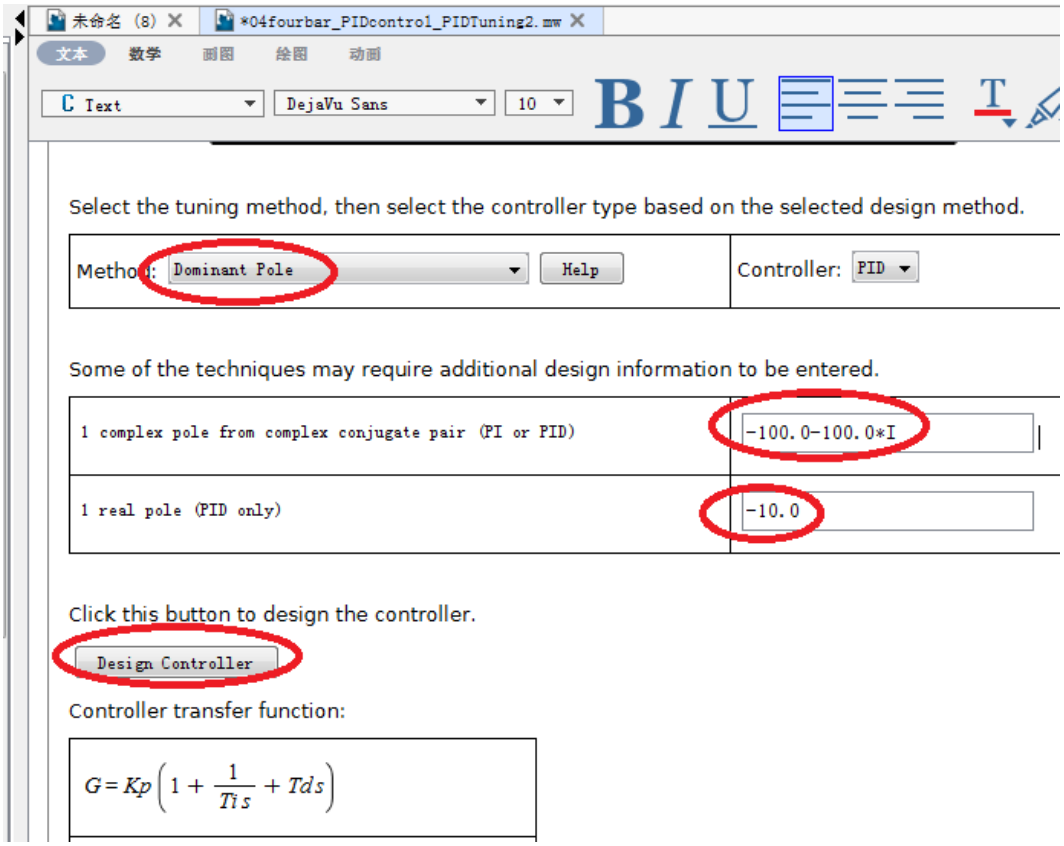


图 2-52　定义参数

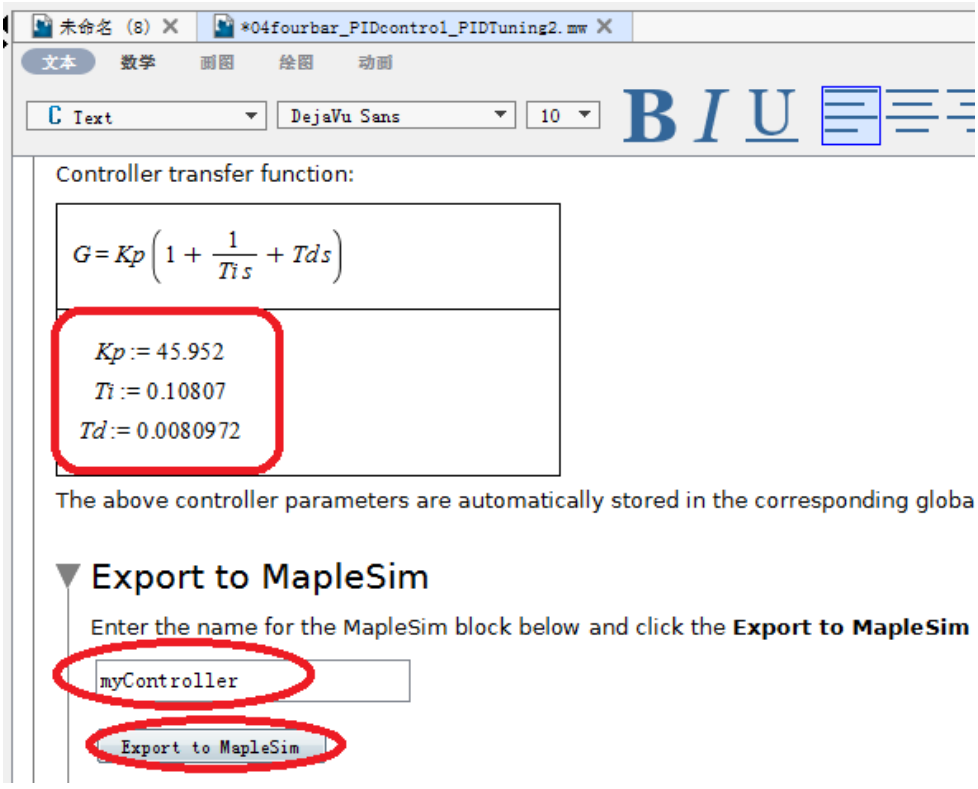


图 2-53　修改 PID 中的参数并导出

3. 用新的 PID 控制器控制四连杆运动，查看控制效果

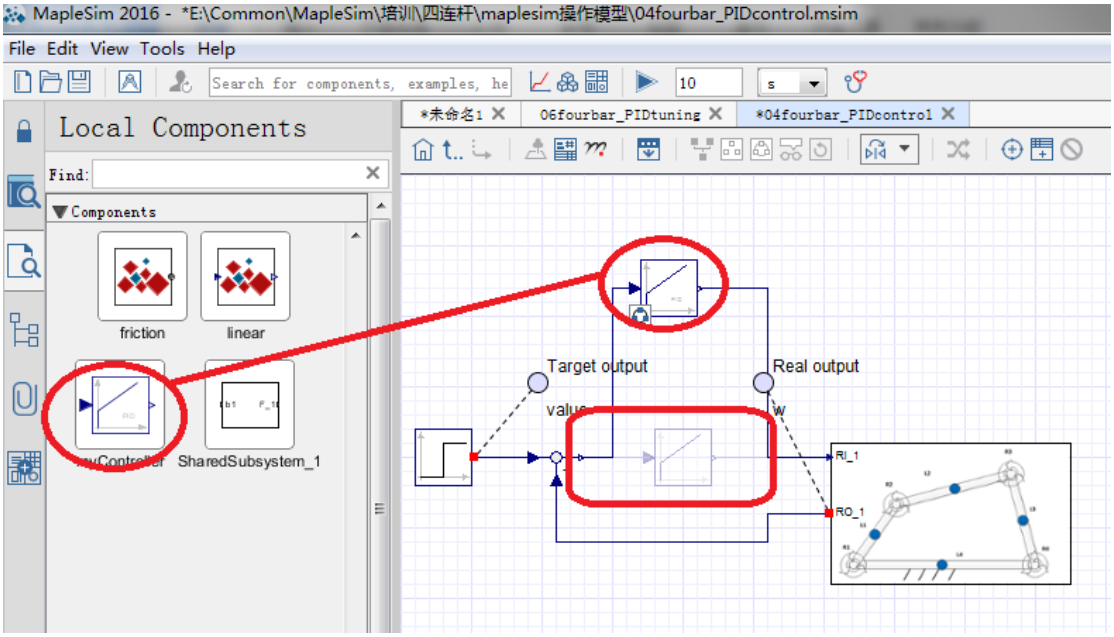


图 2-54　添加新创建的元件

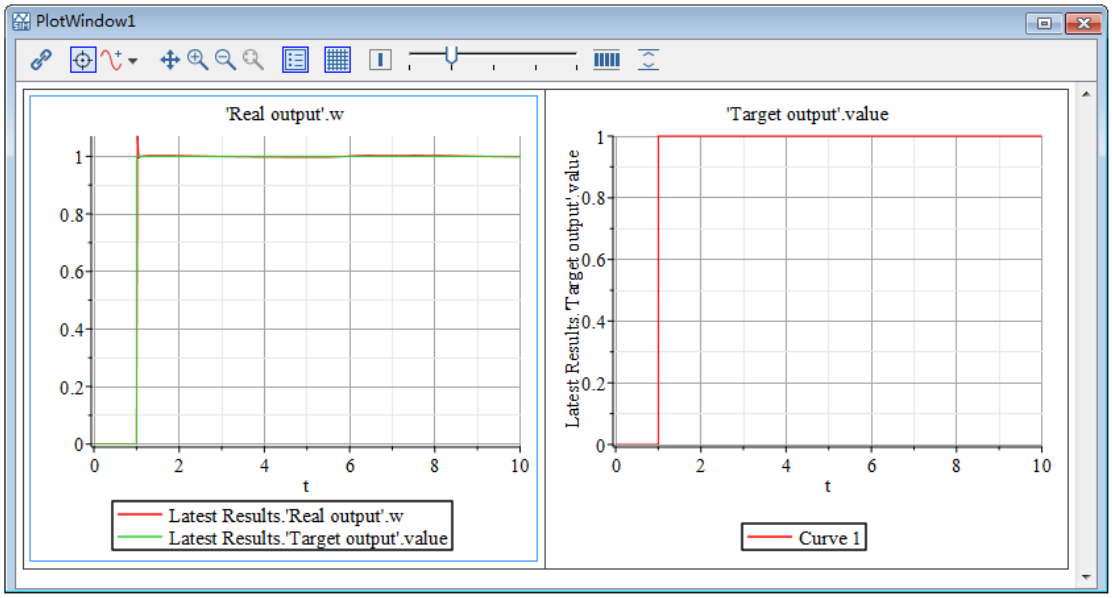


图 2-55　仿真结果图

第三章 CAD 工具箱建模

概要

在本文中，您将会通过一个 Mechanism.step 文件来建立一个曲柄滑块机构。我们将通过以下几步来实现它。

1、导入 CAD 文件；

2、将零部件分组合并到子装配图；

3、添加坐标系并将 CAD 子装配图导入到 MapleSim 中；

4、建立曲柄滑块模型；

5、模型仿真。

模型下载地址：http://www.maplesoft.com.cn/book/3.1.zip

3.1 导入 CAD 文件

1、首先，打开 MapleSim 软件，从文件 **File** 选项中，我们选择导入 CAD（**Import CAD. . .**）。

2、浏览下边的目录，在安装目录下打开文件：

E:\Maple 2016\toolbox\CadToolbox\examples\Mechanism.step。

3、选择 **Mechanism. step** CAD 文件，点击 **Open**。这时 CAD 工具箱窗口打开，同时会显示进度条的进度。一旦进度条完成，装配图的零部件将会显示在（Import CAD）窗口：

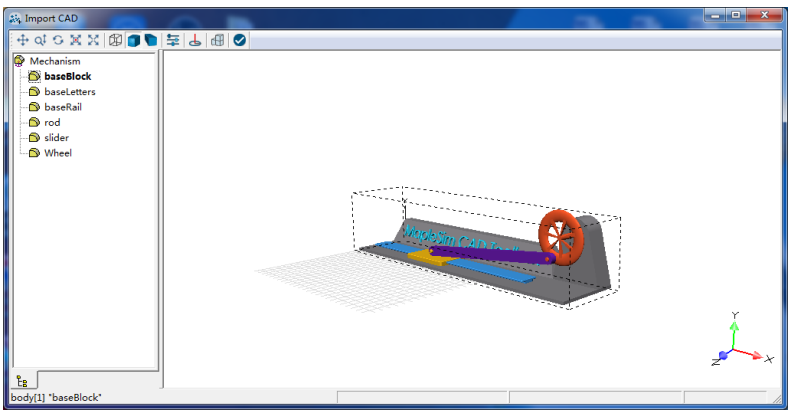


图 3-1　打开 **Mechanism. step** 文件

在 CAD 工作空间使用平移 **Pan**( ), 缩放 **Zoom In/Out** ( ), 转动 **Rotate** ( ), 局部缩放 **Box Zoom** ( ), 查看全局 **View All**( ) 来观看模型。

提示：单击 **ESC** 退出这些查看模式，鼠标重新变回 ( )。

3.2 将零部件分组到一个子装配图

当需要导入零部件到 maplesim 时，分组并合并它们成为一个单独的部件。再装配图中，我们将建立一个包含 baseblock 和 baseletters 的子装配图。

1、在左边的导入 CAD 窗口的 **Object Tree** 中，使用 ctrl+左键选择 **baseBlock** 和 **baseLetters**。

2、右击 **baseLetters**，然后选择 **Create Subassembly**。或者，您可以在工具栏中选择建立子装配图图标 **Creat Sub-assembly**（)。

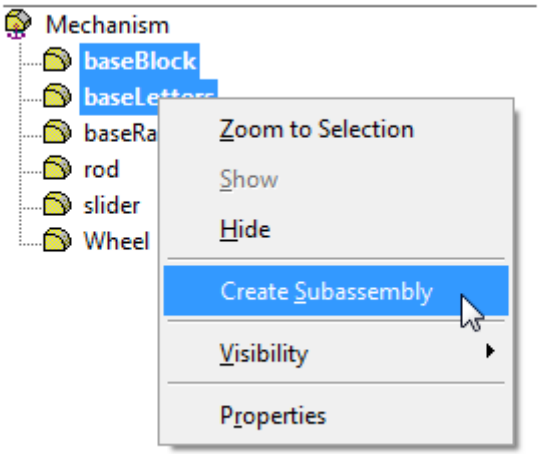


图 3-2　创建子装备图

3、在命名栏 Name 中输入 Base。

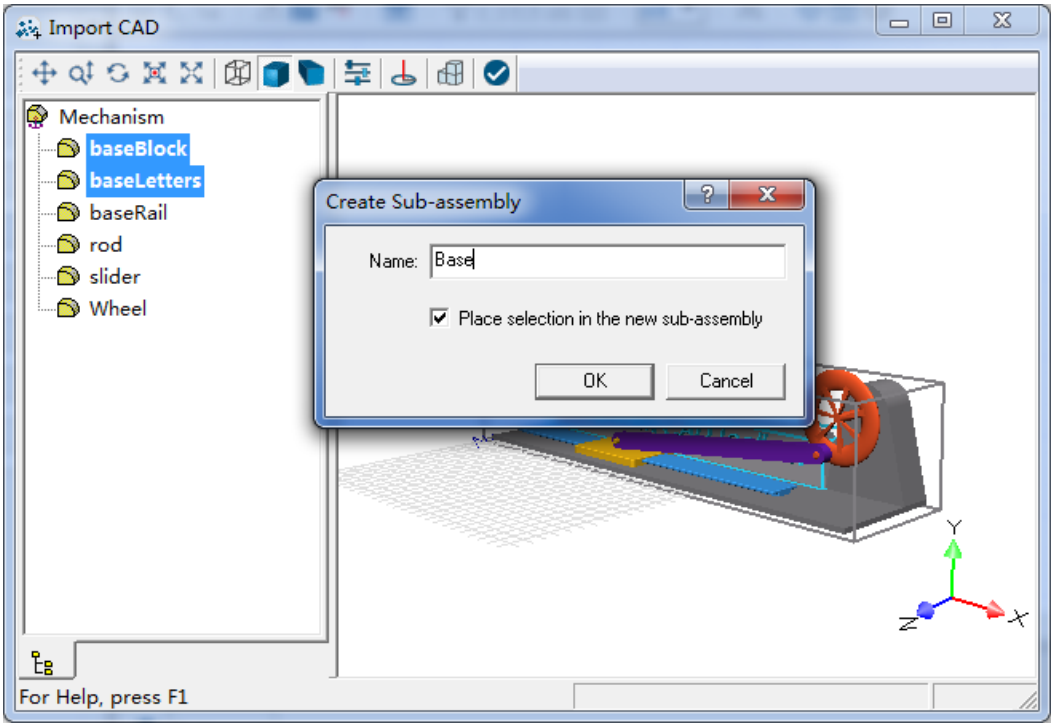


图 3-3　输入子装配图名称

4、点击 **OK**。

更新的对象树中包含了新的 Base 子装配图。Base 包含两部分，baseBlock 和 baseLetters。

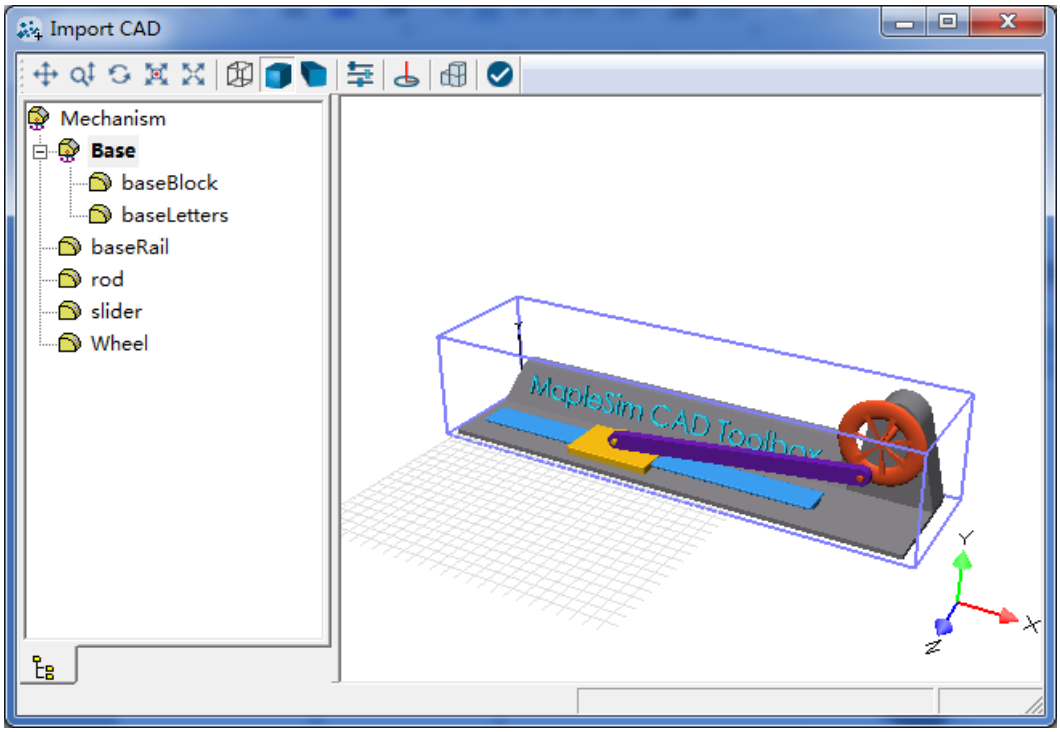


图 3-4　创建完成后的装配图

5、使用左键将 **baseRail** 拖到 **Base** 子装配图中。

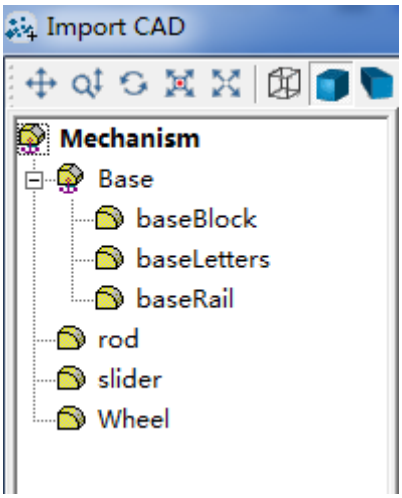


图 3-5　拖入零件后的子装配图

提示：

1、在对象树中，通过拖动一个零件到子装配图，您可以将一个零件添加到子装配图中。同样的，您可以将一个零件移出子装配图，通过拖动一个零件到另一个子装配图或者到主装配图（这个例子的主装配图为 Mechanism）。

2、每个零部件都有可以设置的属性，例如材料（密度）、颜色、透光性。对一个零件设定属性，右击零部件，选择 Properties。对于更多关于零部件属性的信息，查看 Setting Part Properties。

3.3 添加坐标，同时将 CAD 子系统导入到 MapleSim

本节包括了如何准备零部件和子装配图，使它们转变为 CAD 子系统用于 MapleSim 中。

1、点击 **Coord** ()。

根据的模型的大小，可能需要些准备时间。

2、将您的指针放置于**baseRail**的平面上。

当您的指针放置于平面上时，平面的颜色和指针会出现变化。

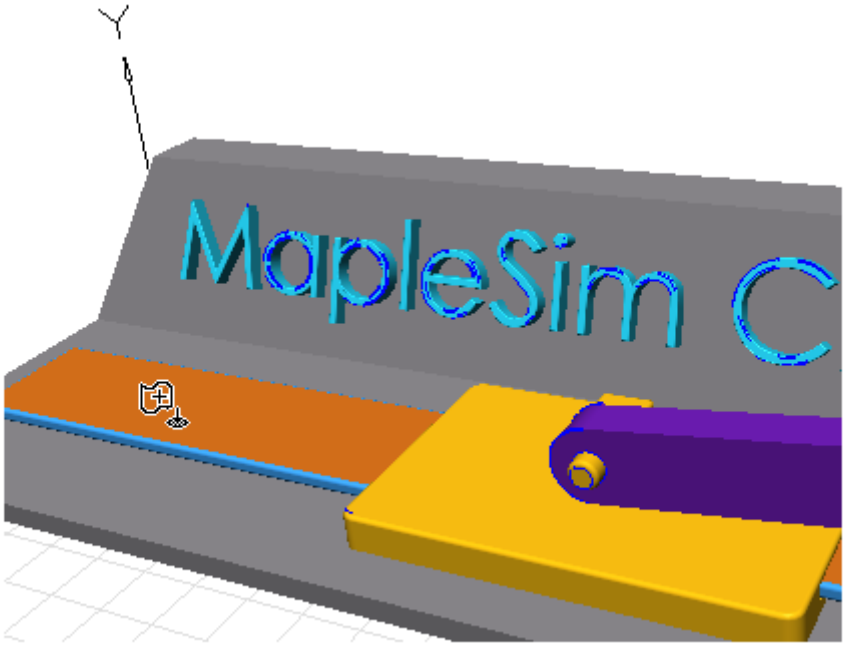


图 3-6　baseRail 平面表面变色

3、点击表面来放置坐标。

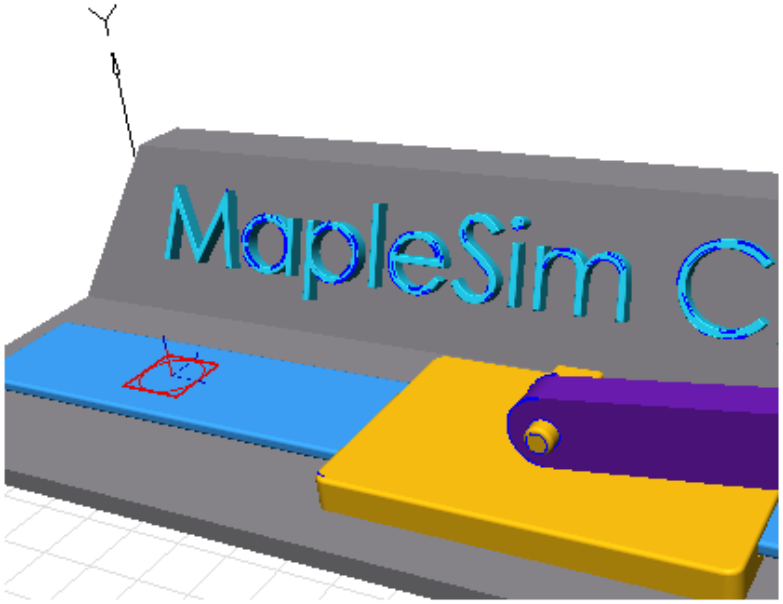


图 3-7　在 baseRail 平面上设立坐标系

4、点击 **Esc** 来退出坐标模式。

5、在对象树 **Object Tree** 中，打开 **baseRail**。

新的坐标就添加到了 baseRail。（仅仅坐标系名字可能不同）

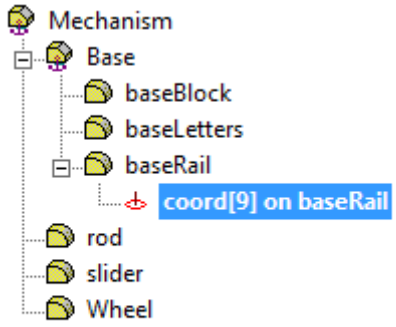


图 3-8　坐标系在装配图对象树中显示

6、在对象树 **Object Tree** 中选择坐标。

7、点击 **Ctrl+C** 复制上边的坐标。

8、在对象树 **Object Tree** 中，选择滑块 **slider**，然后点击 **Ctrl+V**。

另外的一个坐标就复制到了滑块。这两个坐标具有相同的位置和方向。当您转化滑块和基座轨道到 CAD 子系统时，这将产生匹配的端口。

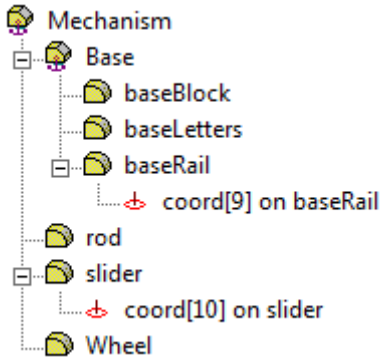


图 3-9　在 slider 元件中添加相同坐标系

注意：在对 **Object tree** 中，您同样可以在两个零部件之间拖动坐标系。

9、通过下列步骤重命名 **baseRail** 的坐标：

A. 选择 **baseRail** 的坐标 `coord[9] on baseRail`；

B. 再点击坐标的名字，就可以编辑名字。在 MapleSim 中，您可以重命名这些坐标名称；

C. 输入 **toSlider** 到命名栏；

D. 点击输入 **Enter**。

10、通过下列步骤重命名 **slider** 上的坐标：

A. 选择附加到 **slider** 的坐标系 ；

B. 点击坐标系的名字；

C. 在命名栏输入 **toBase；**

D. 点击输入 **Enter**。

您的对象树现在如下图：

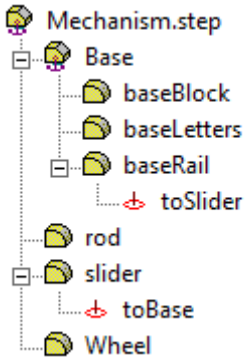


图 3-10　修改坐标系的名称

注意：同样的您可以重新命名坐标通过使用坐标属性窗口。详情请见 Setting Coord Properties。

11、将另外一个坐标附到 **baseBlock** 上。

12、单击 **Esc**。

13、重命名添加到步骤 11 的坐标为 **toGround**。

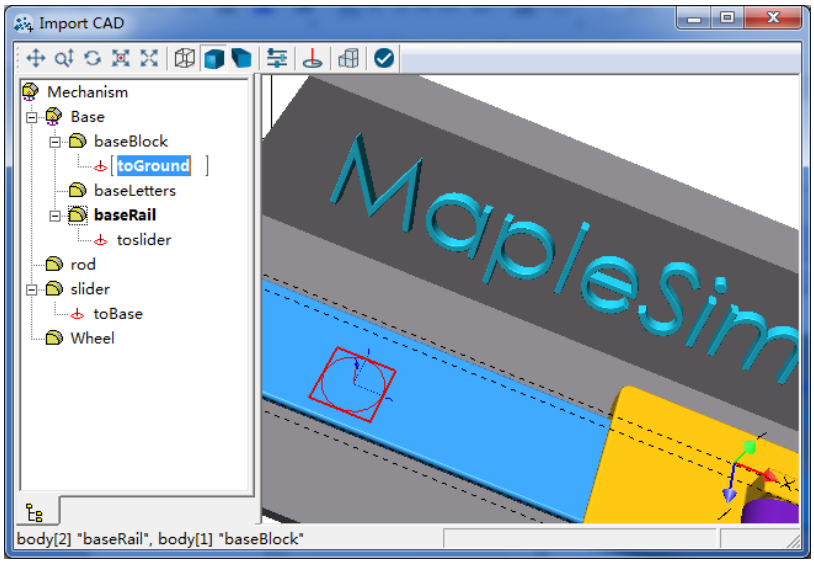


图 3-11　修改 baseBlock 中的坐标系名称

如果您放大坐标，您会发现坐标的 z 轴与方向指示器的 z 轴不一样。

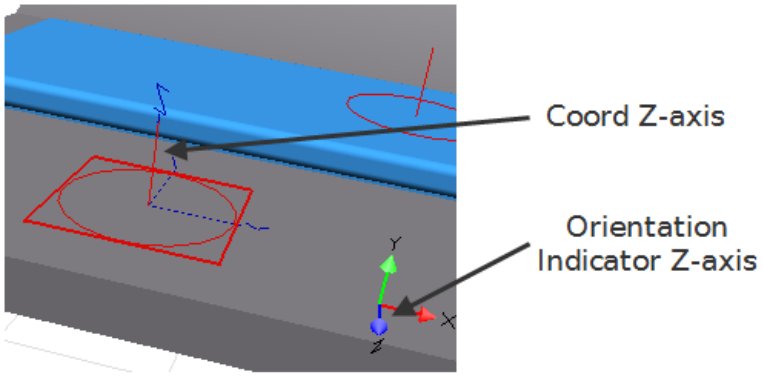


图 3-12　建立的坐标系与方向指示器

坐标的方向决定了零部件的方向在 Maplesim 中是如何决定的。我们调整 toground 的方向，使它符合方向指示器。

14、点击显示设置 **Display Settings** ( )。

15、选择 **Units**，在 **Rotation** 菜单中选择 **deg**。

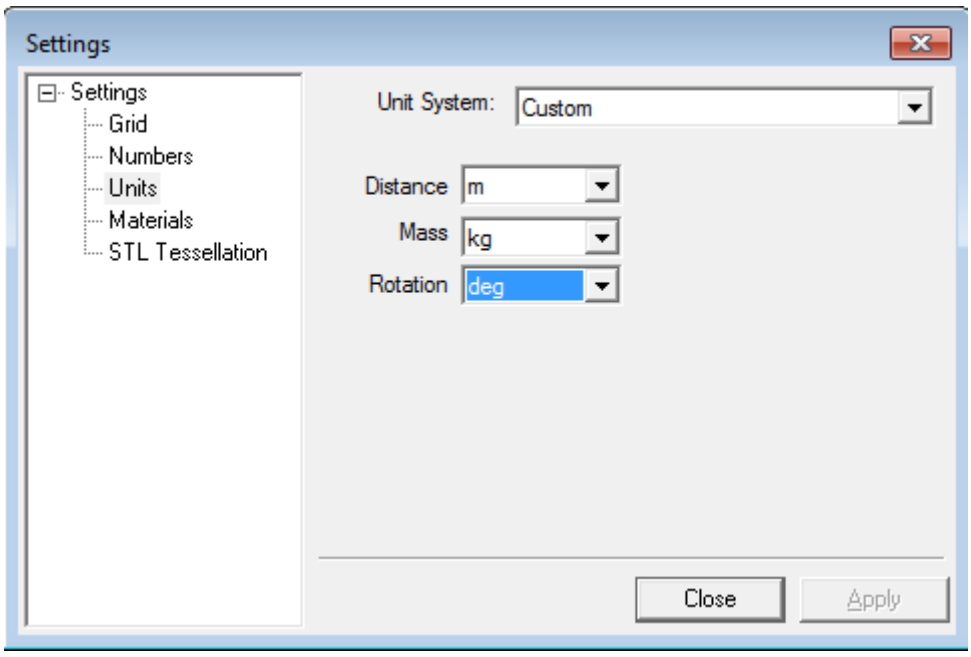


图 3-13　Setting 设置窗口

16、点击关闭**Close**。

17、在对象树**Object Tree**中，右击**toGround**，选择属性（**Properties**）。

18、选择**Pos**，对于**Rx**输入**0**。

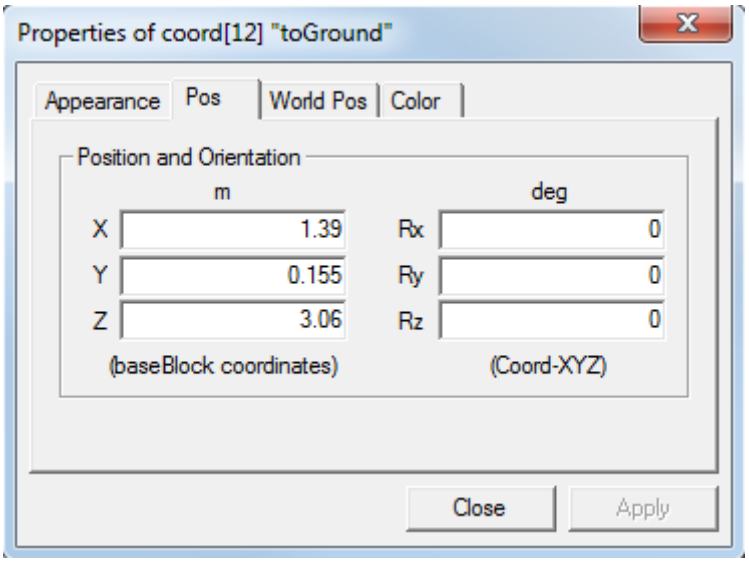


图 3-14　修改后的参数

这项设置将使 toGround 的轴和方向指示器的轴一致。

19、点击关闭 **Close**。

20、Shift+**Z** 得到一个 CAD 装配图关于 Z 轴的视图。

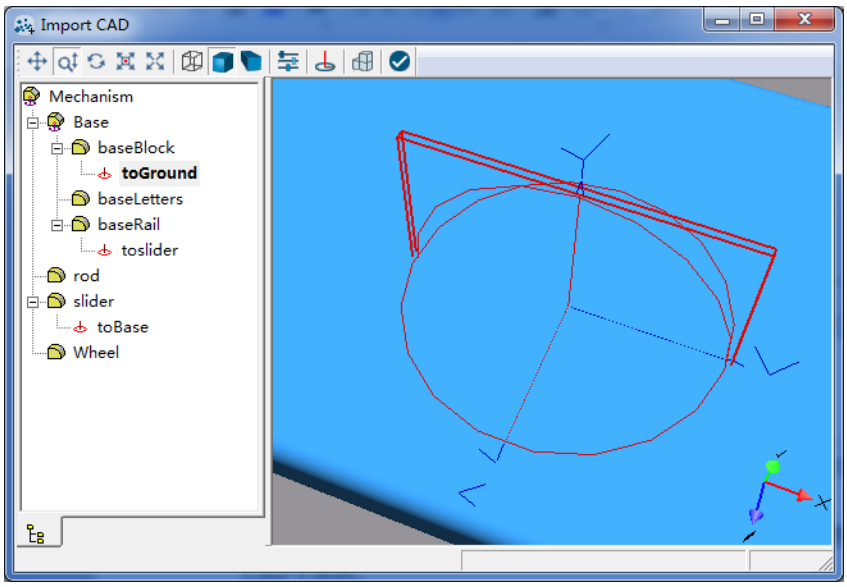


图 3-15　修改后的坐标系与方向指示器方向一致

提示：当前视角在 MapleSim 中将被用作建立 CAD 子系统的图标。一个好的视图会使您的零部件在 MapelSim 中更容易观察识别。当前视图同样被用作在 CAD 子系统图标中来选择端口的位置，一个好的视角会更容易发现端口。

21、点击 **Accept and Return** ( )来完成 CAD 图的导入和转化零部件和子装配

图到 CAD 子系统。

22、点击 **OK**。

这样，CAD 图的零部件就转化为了 CAD 子系统，同时导入到 MapleSim。CAD 子系统可以在左侧项目面板的 **Local component** 的 **Hierarchy** 下边找到。

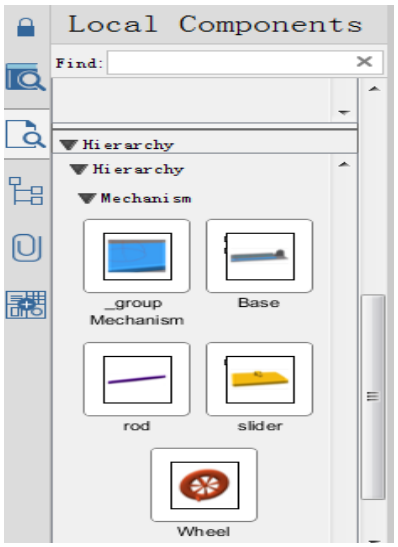


图 3-16　CAD 的子系统元件

需要注意的是，对于在 Import CAD 窗口中的每个零部件和子装配图都有一个 CAD 子系统。同样，这里有一个_group Mechanism 元件。_group Mechanism 元件是一个共享子系统，其中包含了您所建立的所有 CAD 子系统。

3.4 建立曲柄-滑块

在本节中，通过连接在 **Hierarchy** 中的 CAD 子系统到多体元件，您将在 MapleSim 中建立一个曲柄滑块模型。在 Feature Detect 窗口，您将看到如何添加匹配的端口到两个 CAD 子系统。

1、从 Hierarchy 中，拖动一个 Base 元件到模型工作空间。

2、拖动一个 slider 元件到模型工作空间，将它放在 Base 元件上方。

在模型工作空间，CAD 子系统分别被命名为 Base1 和 slider1。注意到两个元件都有一个 CAD 子系统图标 ( )在它们的左下角。这个图标附加到模型工作空间中的所有 CAD 子系统。

提示：如果您没有看到 CAD 子系统图标，点击 **View 菜单**，选择 **Show Subsystem Markers**。

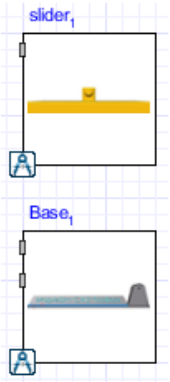


图 3-17　各元件位置及子系统图标

3、在 MapleSim 主工具栏中点击 Show 3-D Workspace ( ) 。

Base 和 slider 元件在 **3−D** 工作空间显示恰当的位置和方向。如果工作区间下方的工具栏中已经选中了 **implicit geometry** ( ) 和 **Toggle axis visibility** ( ),您将会看到由多体端口（坐标轴），刚体结构（圆柱体），和刚体（球体）所组成的 Base 和 slider CAD 子系统。特别的，您可以看到下列特征:

a. toGround 端口；
b. toSlider 和 toBase（匹配的端口）；
c. 从 toSlider 到 Base 的质心的刚体结构；
d. 从 toGround 到 Base 的质心的刚体结构；
e. 从 toBase 到 slider 的质心的刚体结构；
f. Slider 的质心刚体；
g. Base 的质心刚体。

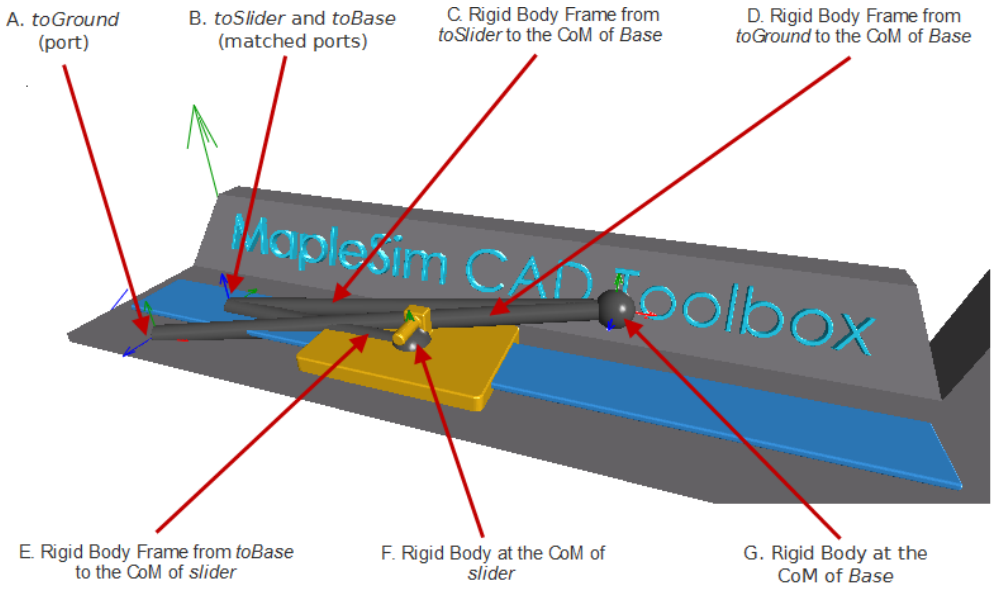


图 3-18　各个零件的坐标系端口及质心

提示：点击显示/隐藏 implicit geometry( ) 来隐藏多体元件。

4、在元件上显示端口标签：

1) 在模型工作空间 Model Workspace，右击 Base1 元件，选择 Show Port Labels。

2) 右击 slider1 元件，选择 Show Port Labels。

端口名字显示在端口旁边。这使得您连接端口时更好识别。注意端口名字和您所给的坐标系的名字是一样的。

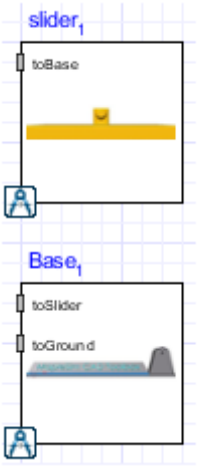


图 3-19　显示端口名称

5、点击 MapleSim 工具箱中的 Show 3-D Workspace ( )。

6、点击 Always-On-Top( ) 。

设置 Always-On-Top 使得 3-D 工作空间已知保持在最前端。对于当您需要在模型工作空间操作同时查看您的模型在 3-D 工作空间装配时是很有用的。

7、放置您的指针在一个 CAD 子系统端口。

这时端口加亮，显示端口的名字。同时，端口的坐标轴在 3-D 工作空间也加亮。

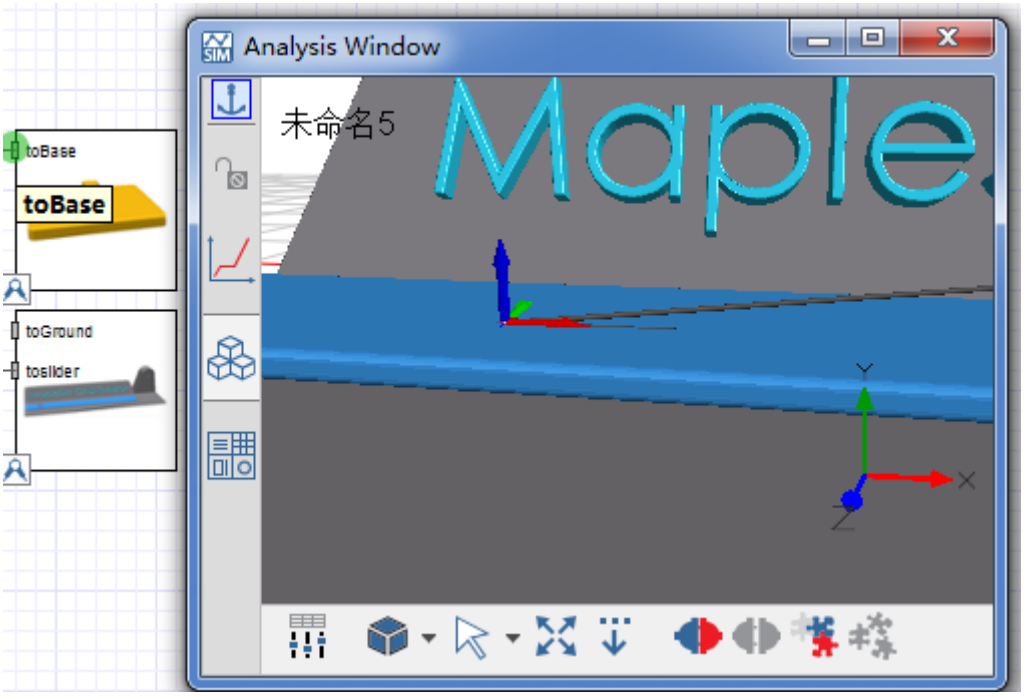


图 3-20　端口在 3D 模型中的显示

8、在 **Libraries Components** 列表中，打开 **Multibody** 〉 **Joints and Motions** 菜单，拖动 **Prismatic** 关节到 **slider1** 左边。

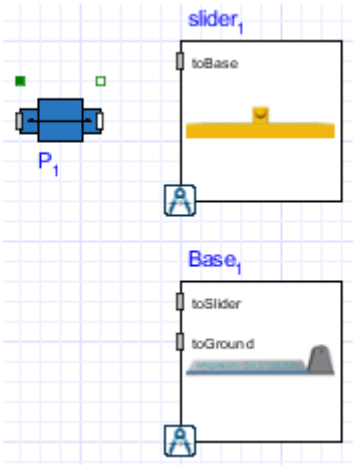


图 3-21　加入移动幅

9、右击 **P1**，选择 **Rotate Counterclockwise**。

10、 **Base1**、**P1**、**slider1** 如下图连接。

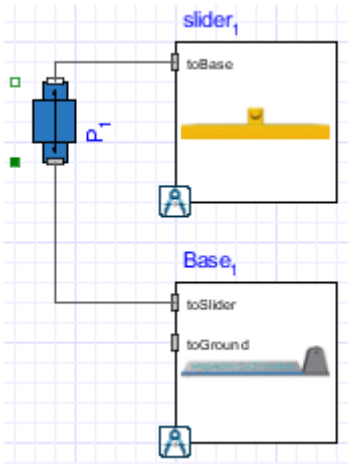


图 3-22　元件之间的连接

注意:步骤 7 中的图显示了 Base 和 slider 之间的滑动轴为 X 轴(红色箭头）。移动关节设置为默认的 X 轴移动，所以不必再设置更改它。

11、在 **Definition** 中，打开 **Hierarchy > Hierarchy > Mechanism** 菜单，拖动一个 **rod** 元件到 **slider1** 右边。

12、拖动一个 **Wheel** 组件到 **Base1** 的右边。

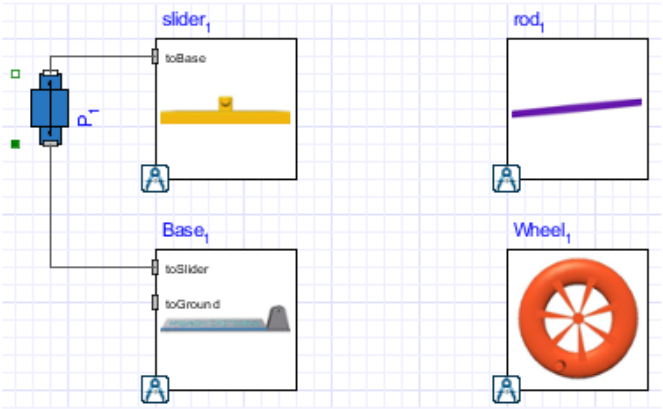


图 3-23　加入其它元件后的模型

13、双击 **rod1**。

在 CAD 工作空间中，添加坐标到这个零件，这样您可以将它连接到其他元件。

14、点击坐标 **Coord** ( )。

15、将您的指针放置在 rod 的右端孔的圆弧上。

当您点到一个圆弧时，您的指针变为十字时，您就会知道（如下图）。同样的，在您处于添加 Coord 模式时，圆弧会自动检测并加亮。

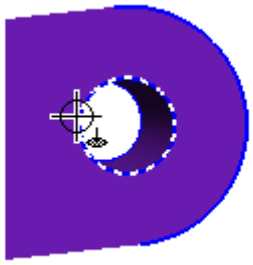


图 3-24　零件模型的圆弧被选中

16、点击圆弧放置一个坐标在孔的中间。

17、重新命名这个坐标为 **toWheel**。

18、放置指针到 rod 的左端孔的圆弧上，点击圆弧。

19、重新命名坐标为 **toSlider**。

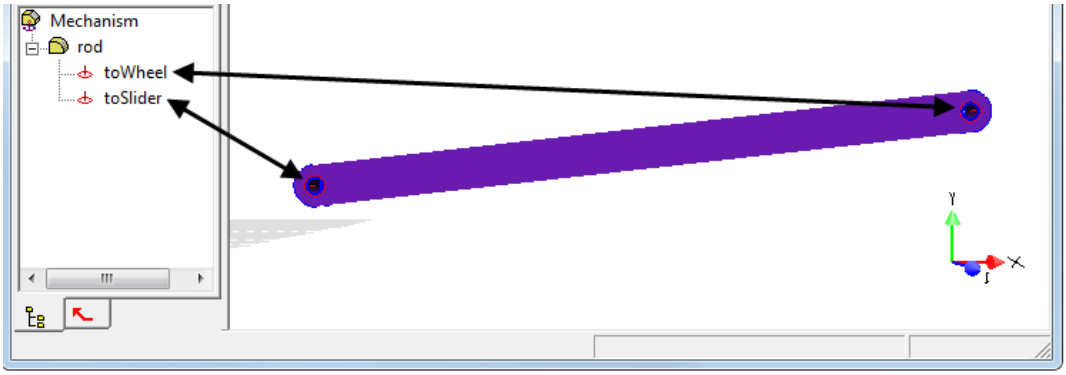


图 3-25　修改零件中坐标系的名称

20、在对象树 **Object Tree** 中，选择 **toWheel** 坐标，点击 **Ctrl+C**。

21、点击 **Accept and Return** ( )，然后点击 **OK**。

22、右击 **rod1**，选择 **Show Port Labels**。

23、双击 **Wheel1** 打开 CAD 子系统。

24、在对象树 **Object Tree** 中，选择 **Wheel**，然后点击 **Ctrl+V**。

25、重新命名为 **toRod**。

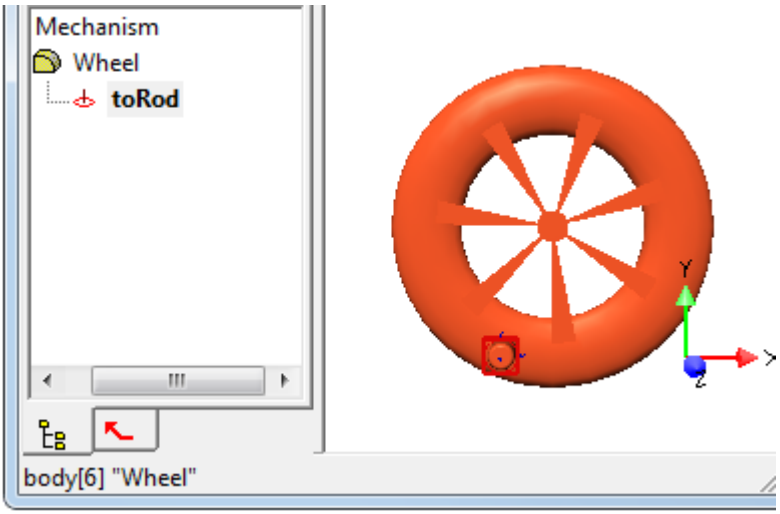


图 3-26　Wheel 零件一端的坐标系

26、点击CAD工作空间的空白地方，然后点击**Shift+X**。（视角变为X轴）

27、点击坐标 **Coord** ( )，在杆的末端圆弧处放置指针（如下图），然后点击圆弧放置指针。

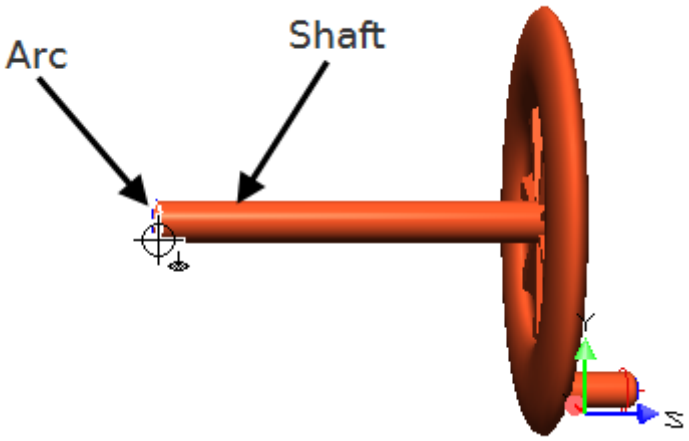


图 3-27　Wheel 零件另一端的坐标系

注意：正确放置的坐标它的 z 轴垂直于杆的末端（如下图）。如果坐标没有正确放置，删除它，重复这个步骤直到坐标在正确的位置和方向。

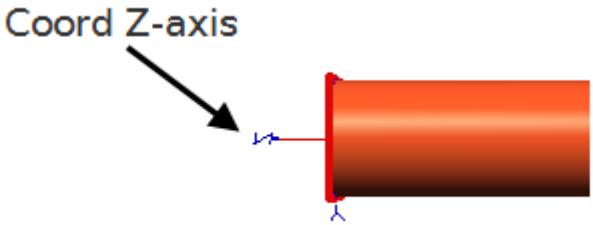


图 3-28　新增坐标系的方向

28、点击 **Esc**。

29、重新命名为 **toBase**。

30、在对象树中选择 **toBase**，点击 **Ctrl+C**。

31、点击 **Accept and Return** ()，然后点击 **Ok**。

32、双击 **Base1** 打开 CAD 子系统。

33、在对象树 **Object Tree** 中，选择 **baseBlock**，点击 **Ctrl+V**。

34、重新命名坐标 **toWheel**。

坐标 towheel 放置于 baseblock 的背面。如果您想查证，转动 baseBlock。

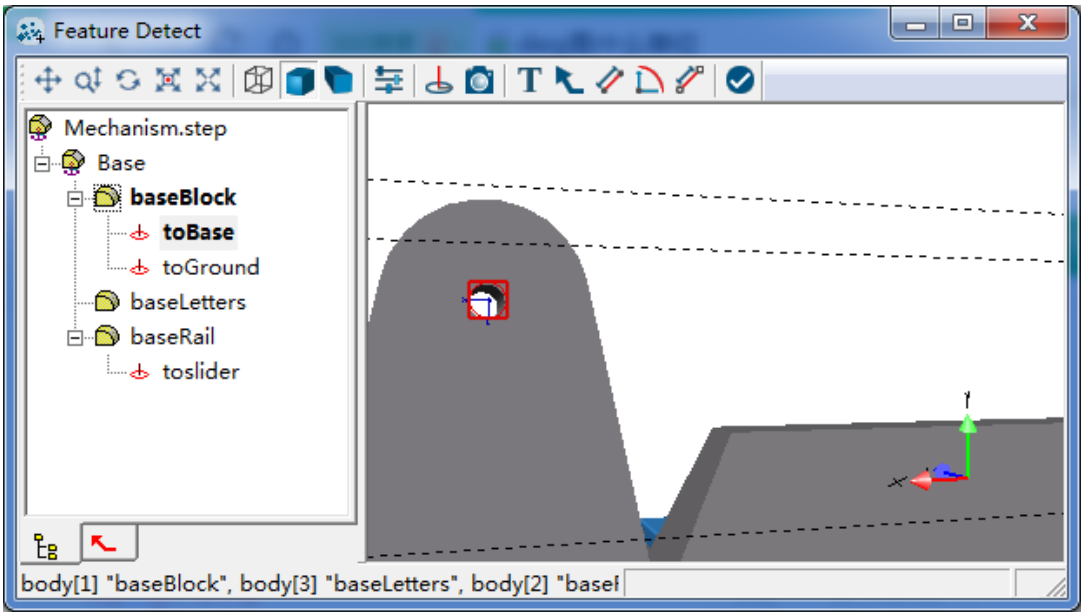


图 3-29　baseBlock 背面的坐标系

35、点击**Accept and Return** ()，然后点击**Ok**。

36、右击**Wheel1**，然后选择**Show Port Labels**。

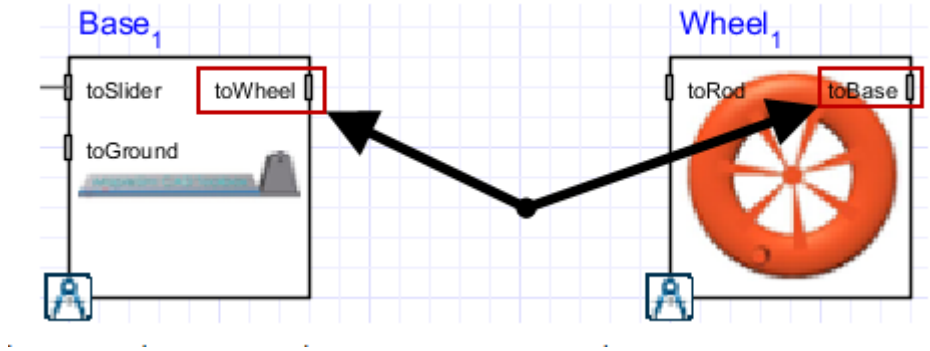


图 3-30　相互对应的端口

37、双击**Wheel1**打开CAD子系统。

38、点击CAD工作空间的空白地方，然后点击**Shift+X**.（视角变为x轴）。

在这个视图中，toBase 在左边，toRod 在右边。这些是在 Wheel1 图标中首选的端口位置。

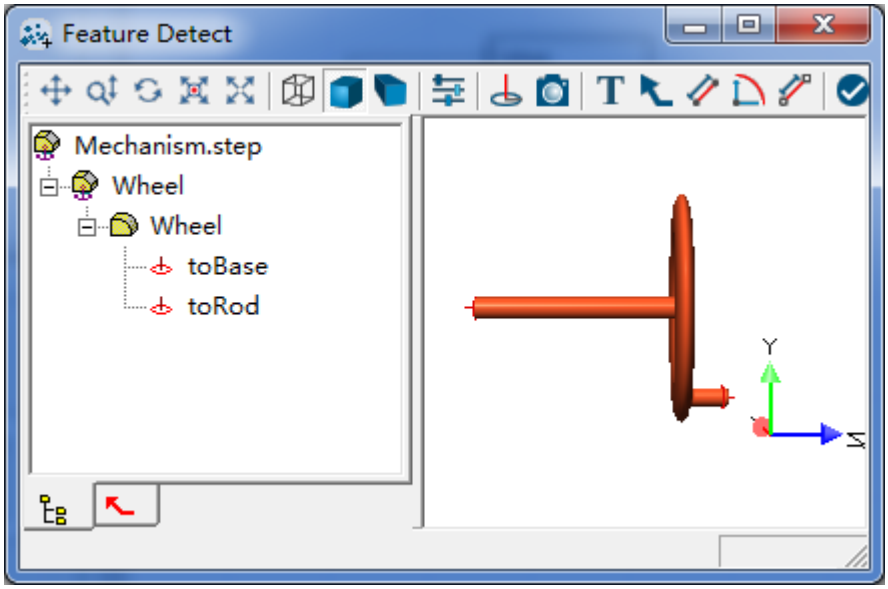


图 3-31　零件中端口位置

39、点击 **Capture Icon Image** ( )（改变子系统图的图案）。

在 CAD 工作空间，捕获的图标图案特征得到一个零件图案，使用该图案作为 CAD 子系统的图标图案。

40、点击 **Accept and Return** ( )，然后点击 **Ok**。

Wheel 的 CAD 子系统图标改变为一个新的图标。toBase 和 toRod 的端口的优先位置变为下图：

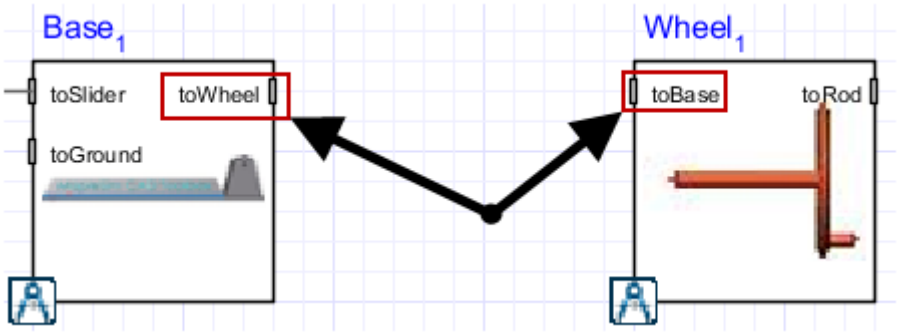


图 3-32　改变图标后对应端口位置

41、双击**Rod1**打开CAD子系统。

42、在对象树**Object Tree**中，选择**toSlider**，点击**Ctrl+C**。

43、点击 **Accept and Return** ()，然后点击 **OK**。

44、在 Feature Detect window 中，双击 **slider1** 打开 CAD 子系统。

45、在对象树 **Object Tree** 中，选择 **slider**，点击 **Ctrl+V**。

46、重新命名为 **toRod**。

47、点击 **Accept and Return** ()，然后点击 **OK**。

所有的组件在模型的工作空间如下图：

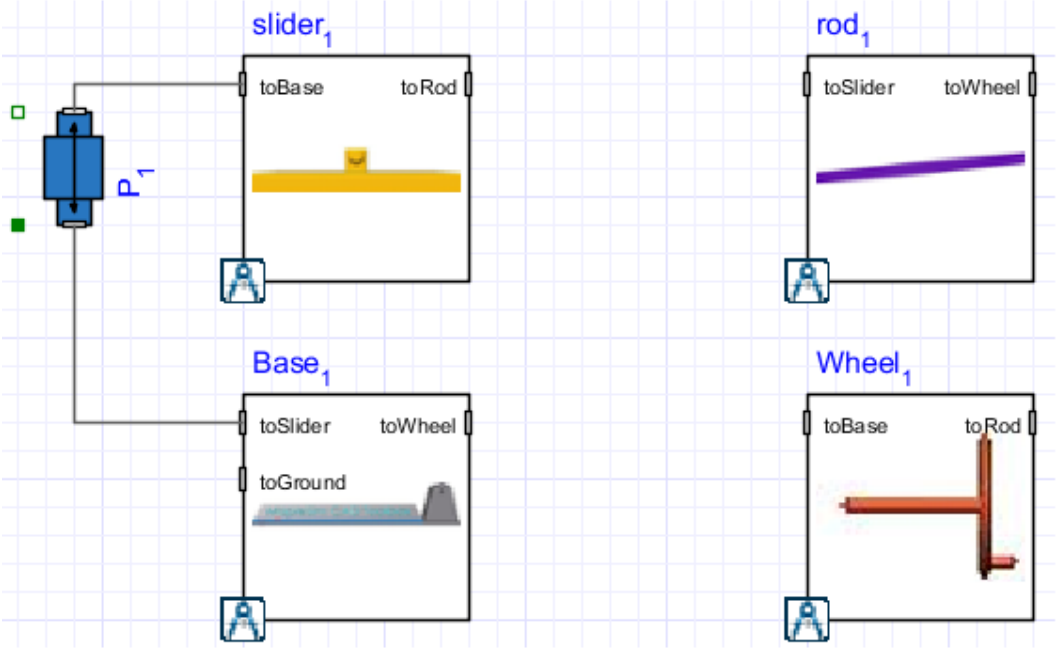


图 3-33　模型的工作空间

48、在 **Libraries** 列表中，打开 **Multibody > Joints and Motions** 菜单，拖动一个转动关节 **Revolute** 到 **Base1** 的右边。

49、拖动第二个转动关节 **Revolute** 到 **Wheel1** 的右边。

50、右击 Wheel1 右边的转动关节，然后选择 **Rotate Counterclockwise**。

51、拖动第三个转动关节 **Revolute** 到 **slider1** 右边。

52、右击 slider1 右边的转动关节，然后选择 **Flip Horizontal**。

53、连接元件（如下图）：

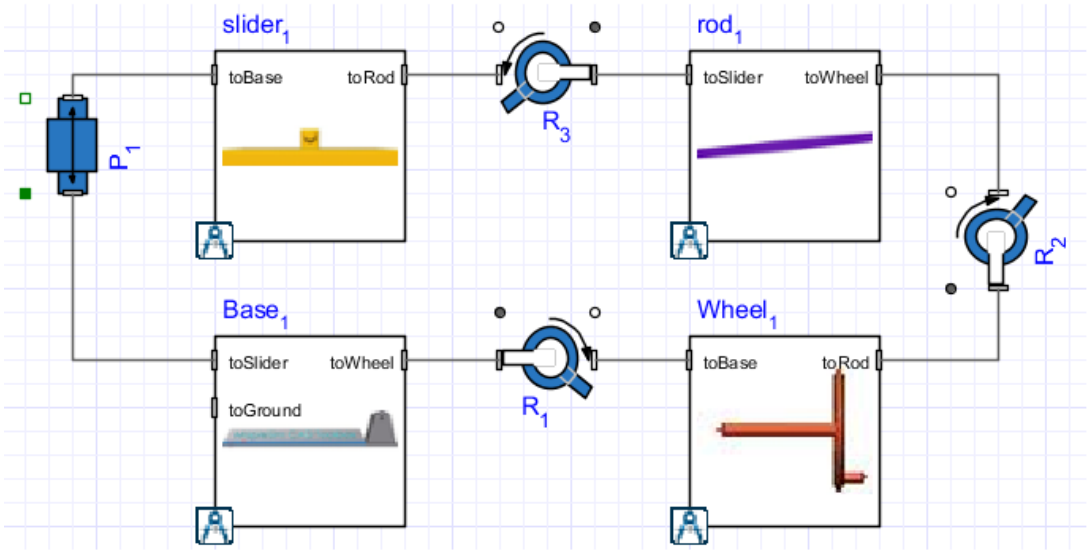


图 3-34　各个元件之间的连接

注意：转动关节都是默认定义绕 Z 轴转动。所有建立的匹配的端口都与它们的 Z 轴一致，并且与 MapleSim 中 Z 轴具有相同的方向。因此，对于刚刚添加的元件无需做任何更改。

54、打开 **Multibody-Bodies and frames** 菜单，然后拖动一个 **Fixed Frame** 组件到 **Base1** 的左边。

55、连接参考坐标系元件到 **Base1** 的 **toGround** 端。

3.5 模型仿真

在对曲柄滑块仿真之前，使用一个恒定速度的元件来驱动曲柄滑块。

1、打开 **1-D Mechanical>Rotational>Motion Drivers** 菜单，在 **R1** 和 **R3** 元件之间拖动一个 **Constant Speed** 元件。

2、连接恒定速度元件的 **flange-b** 到 **R1**。

下图显示了所有端口连接后的完整模型。

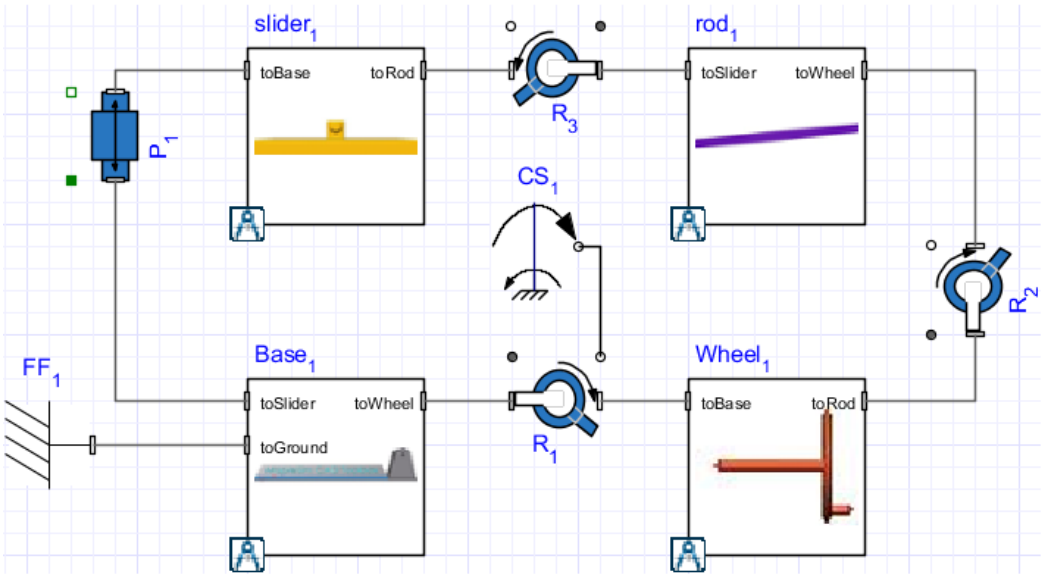


图 3-35　连接后的模型

3、点击 **Run Simulation** ()。

当仿真完成时，在结果管理器中将会出现仿真结果列表。您可以在 3-D 播放窗口运行一个曲柄滑块模型的 3-D 动画。

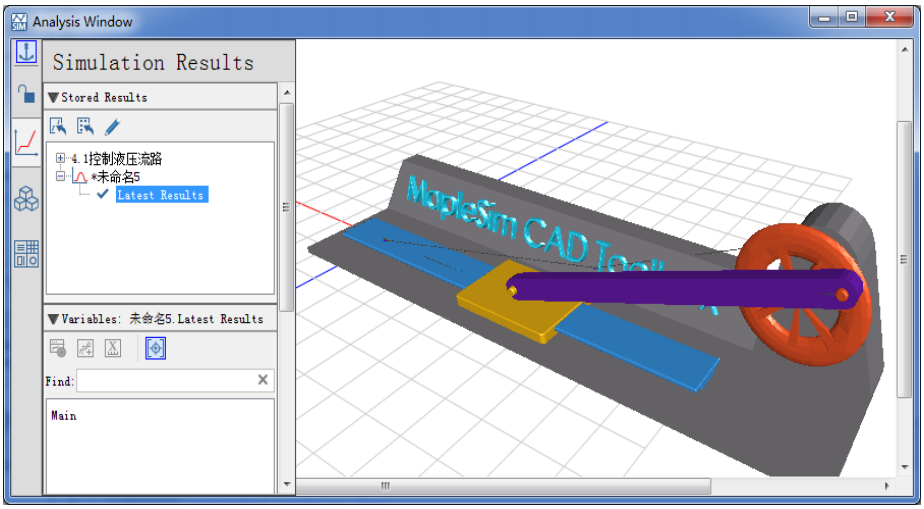


图 3-36　3D 动画演示

第四章 传动系统库建模

4.1 MapleSim 传动系统建模

汽车制造商努力提高燃油效率，主要焦点是汽车动力总成中发动机的功率损耗，但是事实表明传动损耗了很多功率。现在工程师正在付出更多的努力正确评估究竟有多少功率损耗，采取什么样的措施减少损耗，提高整车的效率。因此，汽车传动系统行业积极投身于优化已有的设计和探索新的创新，这些工作基于在计算机中使用传动系统的虚拟样机。虚拟样机通过允许工程师在物理原型前解决设计问题，大幅度降低成本，生成更高效的产品。工程仿真产品，例如 MapleSim，已经成为这个过程中的关键工具，为越来越多的传动系统部门所采用。

Maplesoft 开发了传动系统库， 提供基本元件库和传动系统部件装配，以及完整的动力总成应用案。

模型下载地址：http://www.maplesoft.com.cn/book/4.1.zip

1、发动机

受控扭矩驱动器模型是一个非常简单的模型，常用于仅考虑扭矩/功率输出的情况。目的是提供 IC 发动机（电火花点火、柴油等）的一次近似模型，从而可以驱动传动的其余部分。它使用了一个简单的 PID 控制器，输出需要的扭矩实现期望的速度。该转速可以是传动轴的旋转速度或者是车辆的速度。发动机功率特征是由最大功率/扭矩曲线 vs. 发动机速度的查询表决定的，用于限制驱动轴上的扭矩输出。

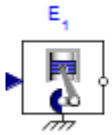


图 4-1 发动机元件

理想发动机输入量为油门开闭，范围为 0（空闲）~1（最大扭矩）。给发动机添加油门输入量，并且加入转动惯量。给转动惯量一个初始速度，作为发动机启动速度。

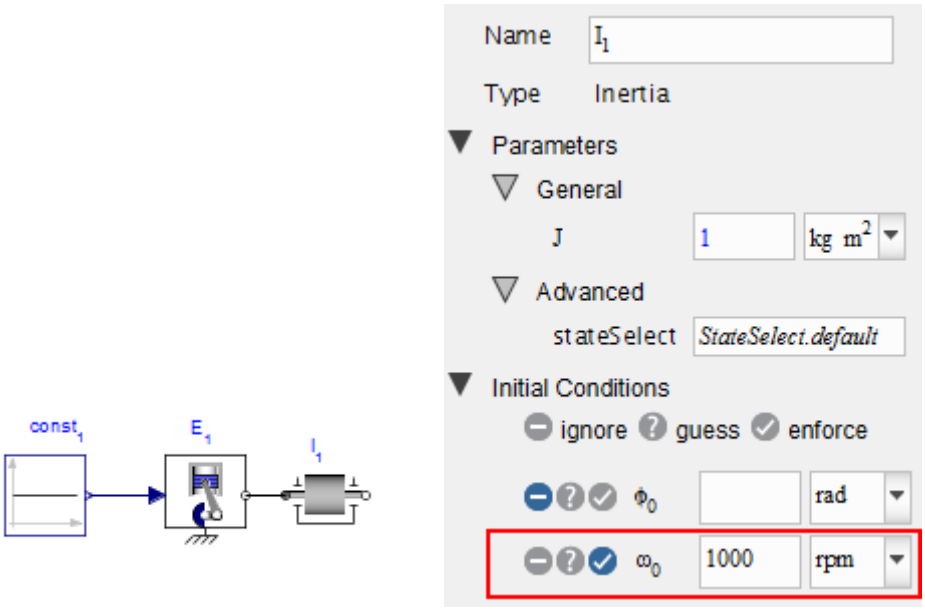


图 4-2　设定惯性元件的初始条件

为测试发动机转速，我们在模型中加入角速度传感器，将传感器输出数值单位设成 revolution per minute，并利用探针测量发动机转速，运行模型得到仿真结果。

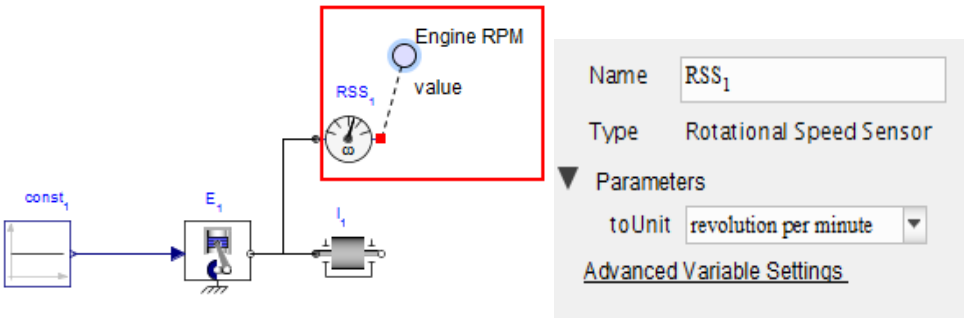


图 4-3 添加元件和探针

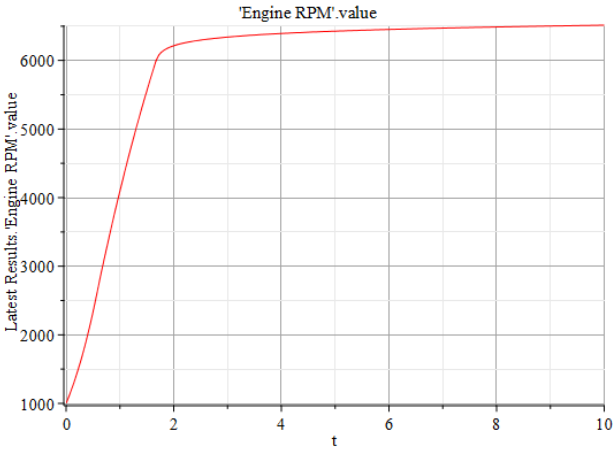


图 4-4 仿真结果图

2、液力变矩器

液力变矩器（液力传动装置）主要用于提供发动机和自动变速装置之间的液力偶合器，当然该机构还有其他的一些应用。它包括在发动机侧有泵轮推进器，在变速器侧有涡轮推进器。在运行时，泵轮驱动流体进入到涡轮推进器，驱使它从发动机扭矩到转动装置。它也包含在低速时放大输出扭矩到变速器的属性，目的是当车辆在停止不动时克服摩擦和惯性。典型地，液力变矩器模型通过变矩比(τ_0/t_i)，和负载能力 vs.速率比(ω_0/ω_i) 的查询表实现。这些特性将随着扭矩流而变化，或者是由发动机驱动 $(\omega_0/\omega_i < 1)$，或者是由传动驱动（当车辆滑行时$\omega_0/\omega_i > 1$）。

图 4-5 液力变矩器元件

3、变速器

Ravigneaux 配置是一个基本的自动变速器装置，它使用一个大的中心齿轮和一个小的中心齿轮，一个支架通过两个独立的星星齿轮连接它们。外侧和内侧的行星齿轮分别围绕支架旋转，但以一个固定的齿轮比逆向旋转。内侧的行星齿轮与较小的中心齿轮啮合，外侧的行星齿轮与较大的中心齿轮啮合。环形齿轮接着与外侧的行星齿轮啮合，并将齿轮和轴耦合在一起。这个配置使用了三个行星-行星齿轮元件和一个行星-环形齿轮元件构成，连接如图所示，提供机械端口：

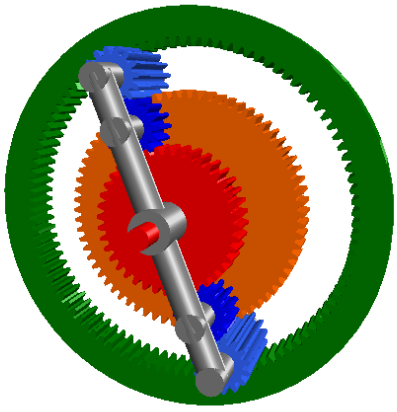


图 4-6 变速器模型

需要给出下面的参数：

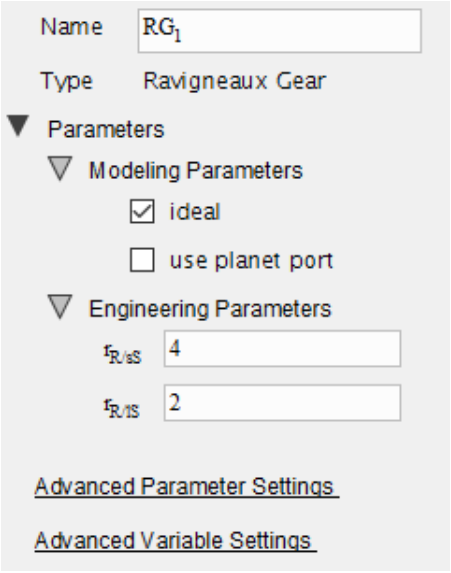


图 4-7 设置参数

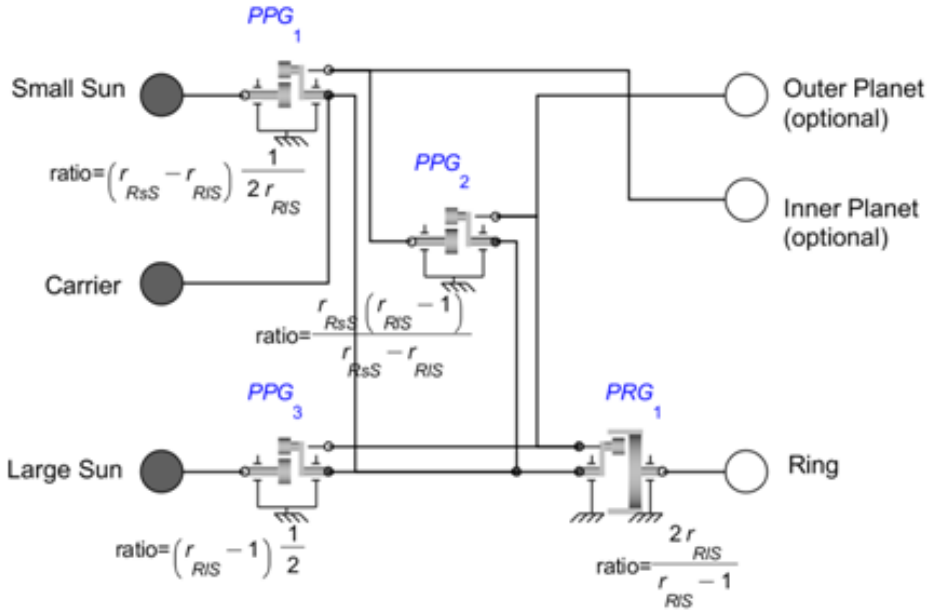


图 4-8 各个元件之间的连接

需要的齿轮比的选择可以通过耦合和分隔机械端口与其他轴或齿轮箱体的连接，借助于离合器元件。

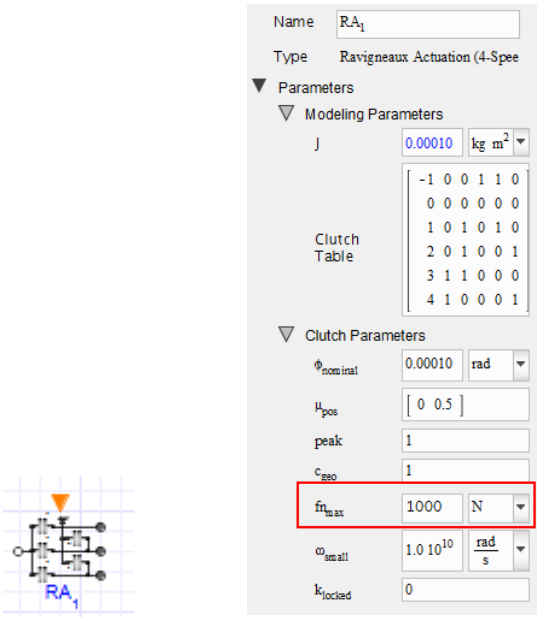


图 4-9 设置 RA_1 的参数

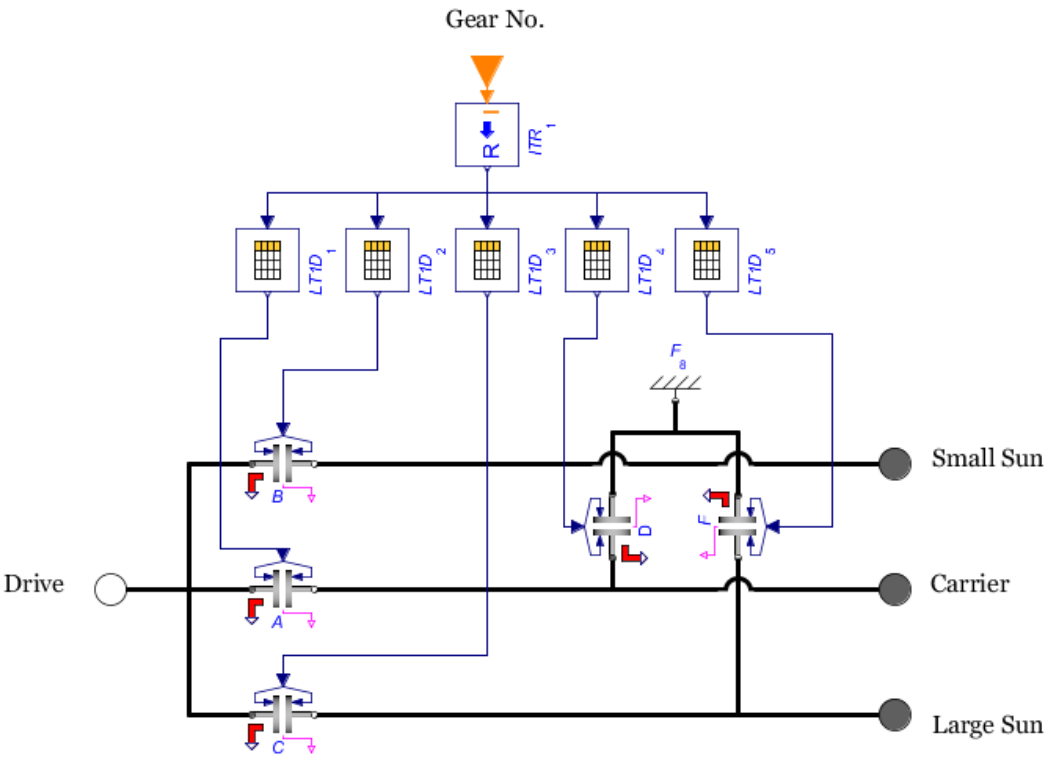


图 4-10 各个元件之间的连接

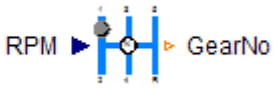
每个齿轮的离合状态，以及由此产生的齿轮比，其档位选择如图所示：

Gear	Clutch States					Ratio Expression	zR=150 zLS=75 zSS=50
	A	B	C	D	E		
R	0	0	1	1	0	$-\frac{zR}{zLS}$	-2
1	0	1	0	1	0	$\frac{zR}{zSS}$	3
2	0	1	0	0	1	$\frac{zR}{zSS}\cdot\frac{(zSS+zLS)}{(zLS+ZR)}$	1.67
3	1	1	0	0	0	1	1
4	1	0	0	0	1	$\frac{zR}{(zLS+zR)}$	0.67

图 4-11 各个齿轮档位的选择

4、自动档位选择器

自动换挡器元件可以应用在各种变速箱中。当发动机转速小于最小转速，档位将减1；当发动机转速大于最大转速，档位将加 1。将自动换挡器与变速箱连接。



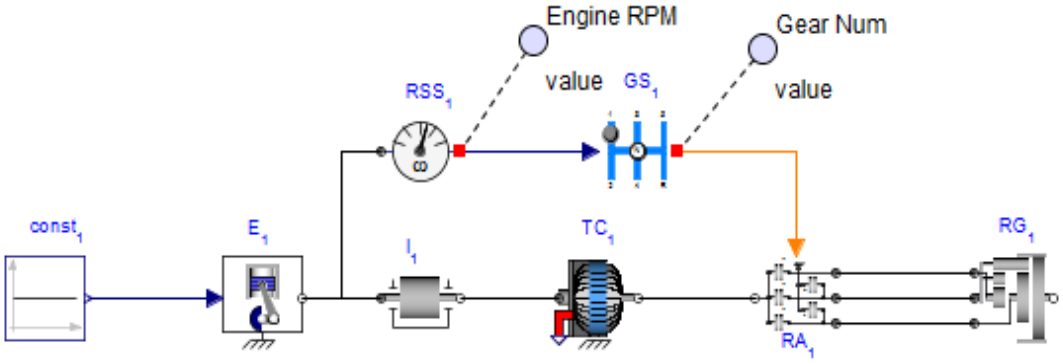


图 4-12 自动档位选择器元件

5、主减速器与负载

在变速箱后连接主减速器，齿轮比为-4。

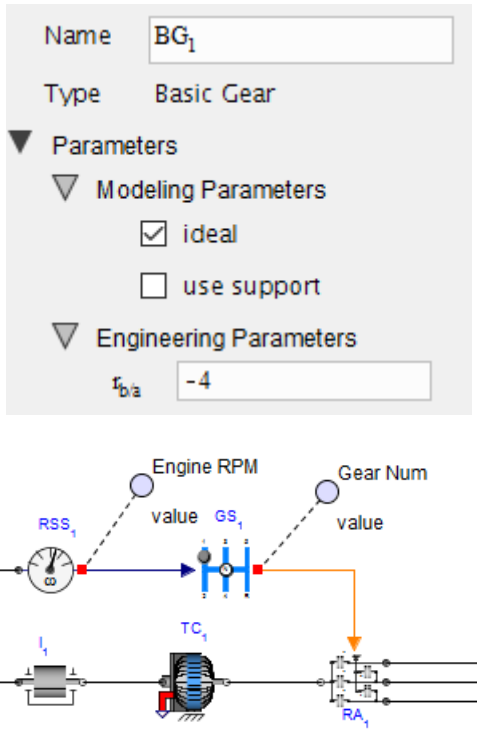


图 4-13 添加齿轮元件并设定参数

主减速器与测功机相连。测功机参数如图所示。

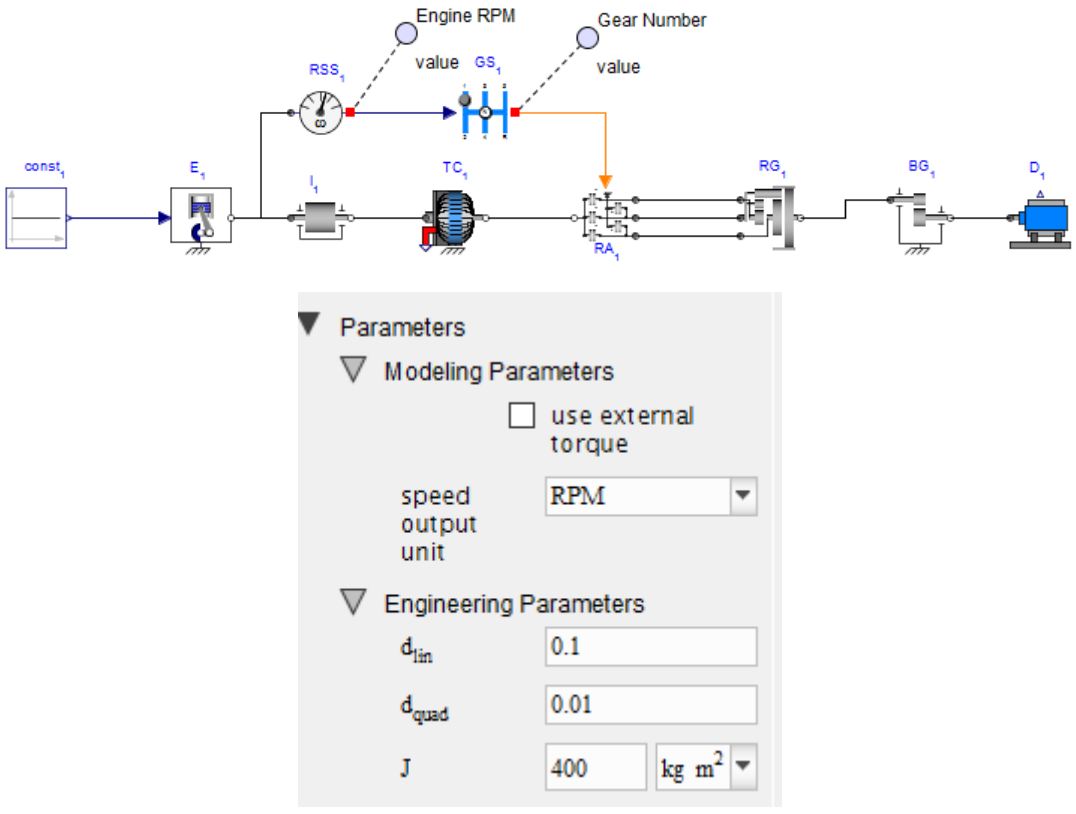


图 4-14 添加测功机元件并设置参数

加入探针，测试测功机输出角速度，单位为 RPM。

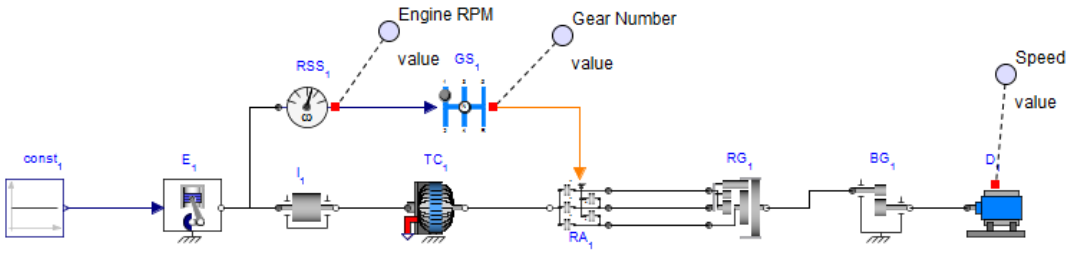


图 4-15 添加探针

设置仿真时间为 150s，运行模型，得到输出结果。

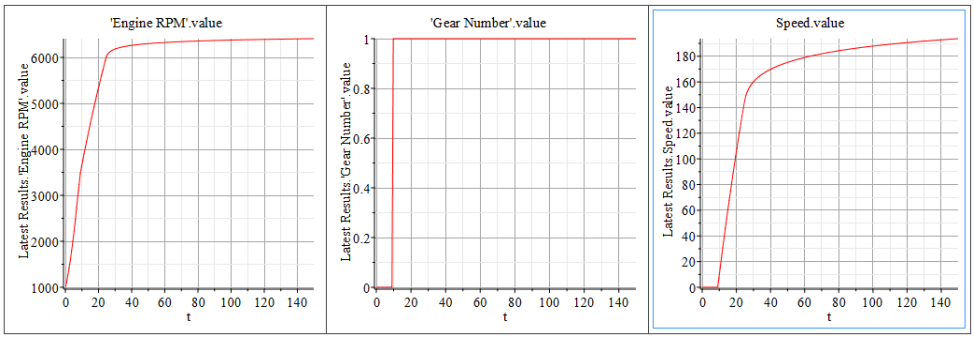


图 4-16 仿真结果图

6、加入 PID 控制

在测功机参数设定中的 use external torque 前打上对号。用外部的 real signal 输入给测功机，提供一个负载扭矩。

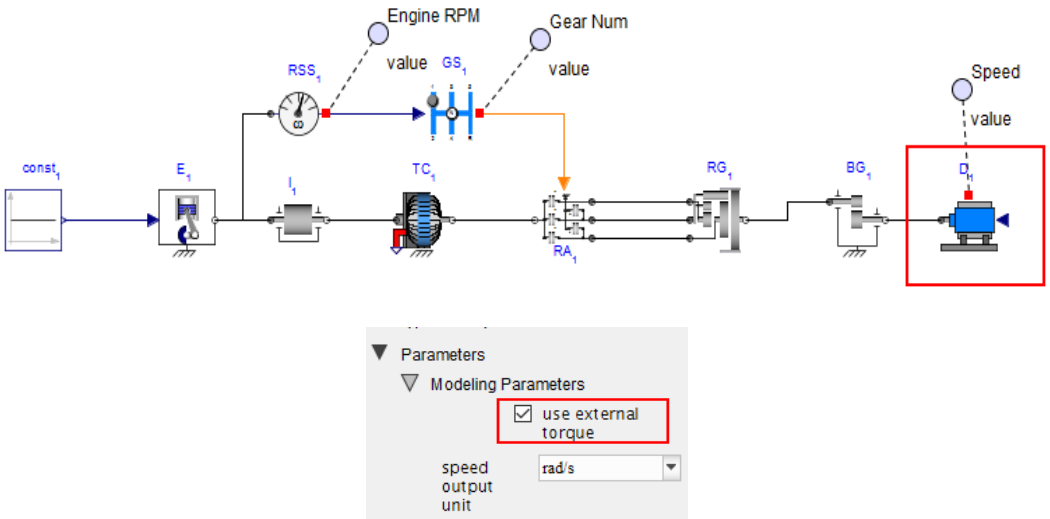


图 4-17 测功机使用外部转矩

删去发动机元件前的 Constant 元件。

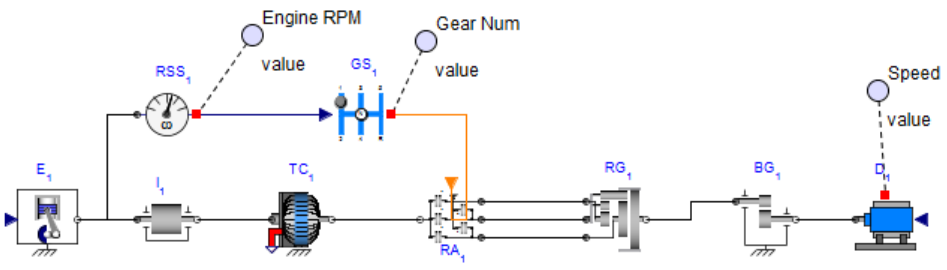


图 4-18 修改后的模型

Ramp 元件和 And 元件设置理想车速。

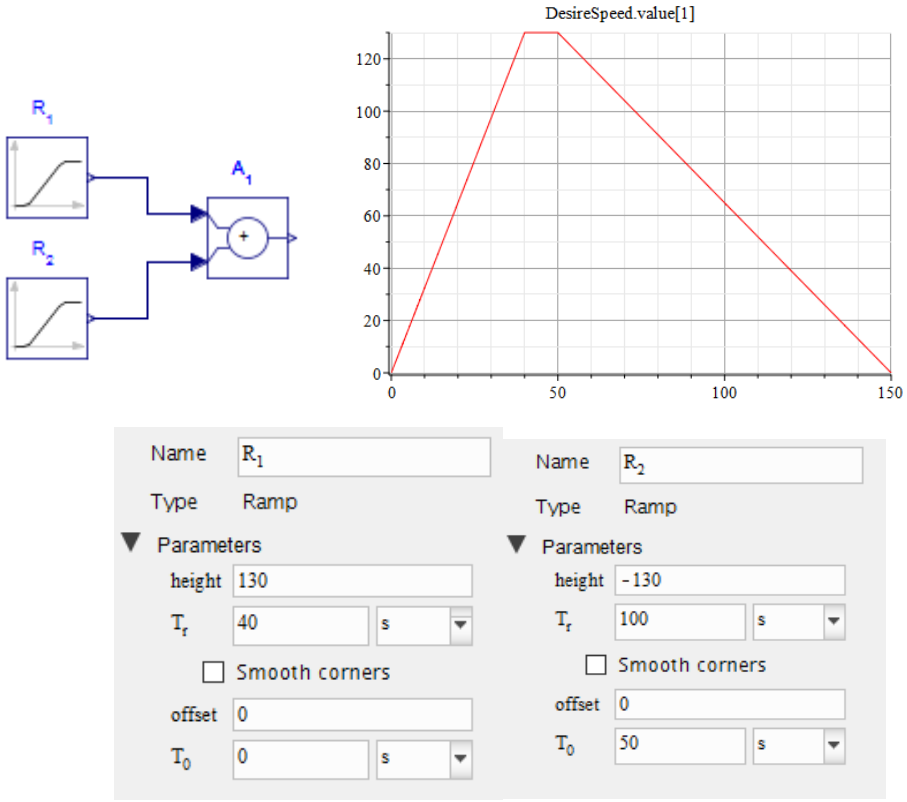
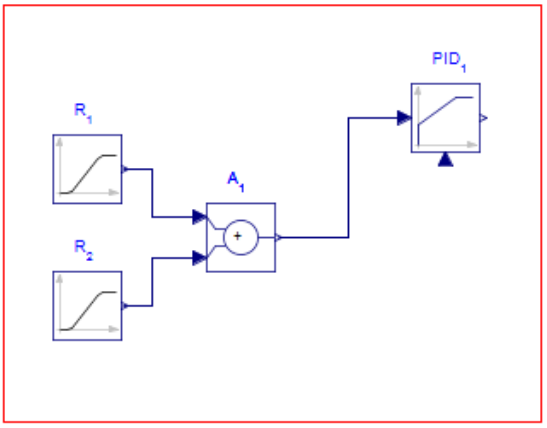
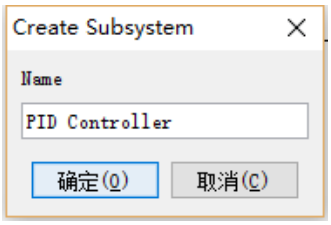


图 4-19 设置 Ramp 和 And 元件的参数

加入 Lim PID 控制元件，并将下图元件设为子系统。按 Ctrl+G 创建子系统，将子系统命名为 PID Controller。





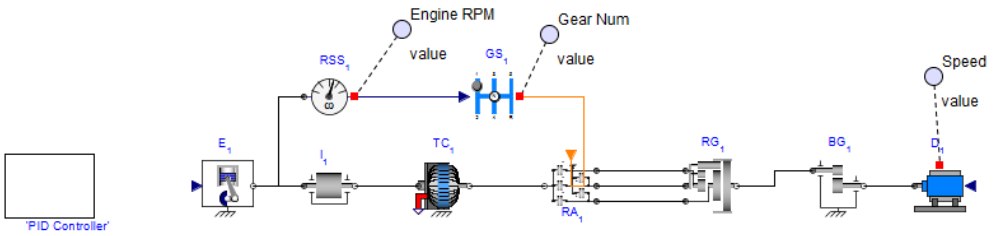
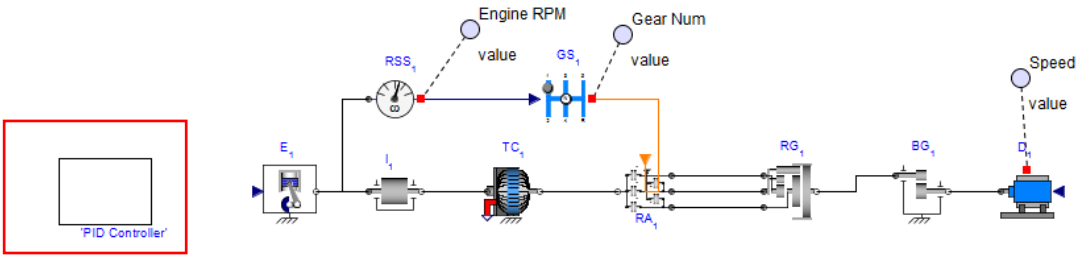


图 4-20 创建 PID Controller 子系统

点开 PID Controller 子系统，加入另一个 Lim PID， 在 Lim PID 右侧加入 Gain 元件。如图所示连接各项元件。



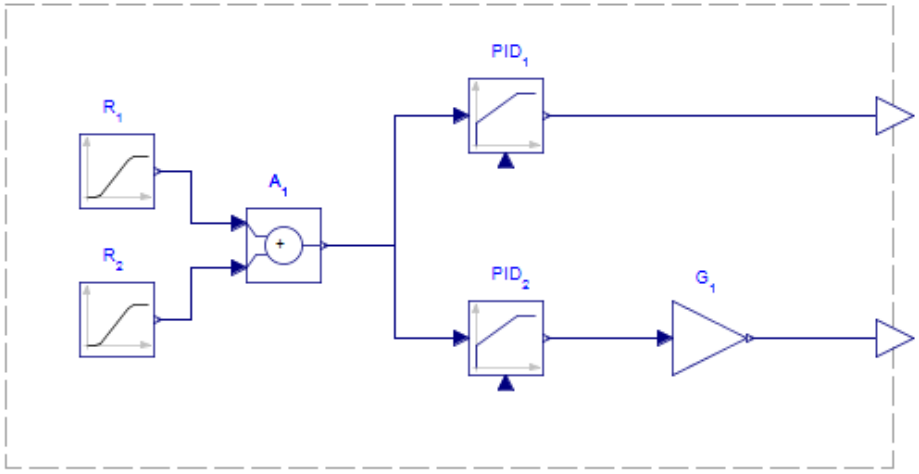


图 4-21　PID Controller 子系统

如下图所示设定 PID 控制元件和 Gain 元件的参数。

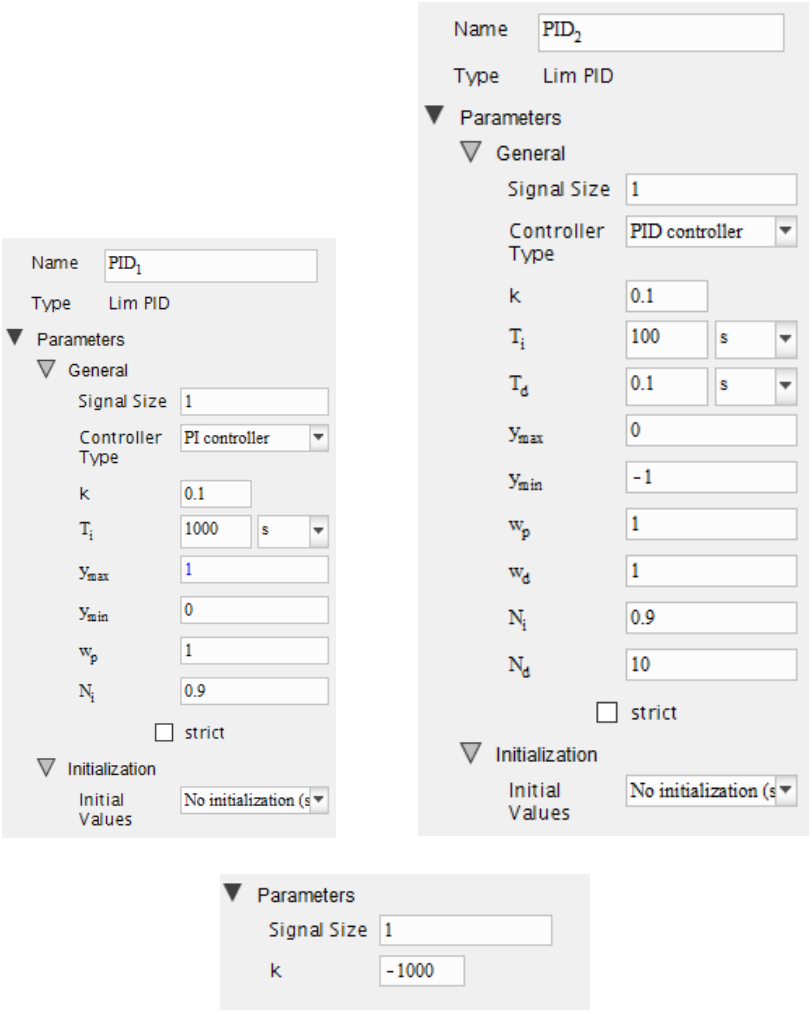


图 4-22 设置 PID 控制元件和 Gain 元件的参数

将 PID 反馈量作为子系统的输入量，将输入量与 PID 控制元件连接。加入探针，测试理想速度，加速度信号和刹车信号。

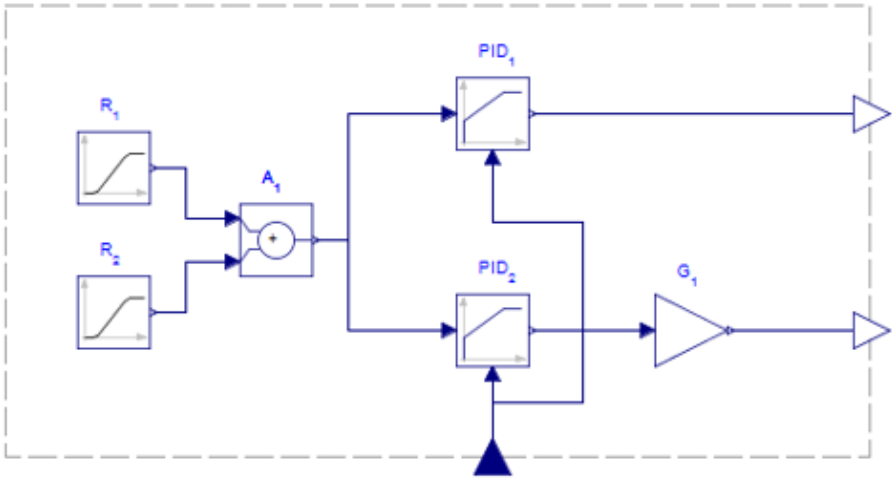


图 4-23　PID Controller 子系统中的连线

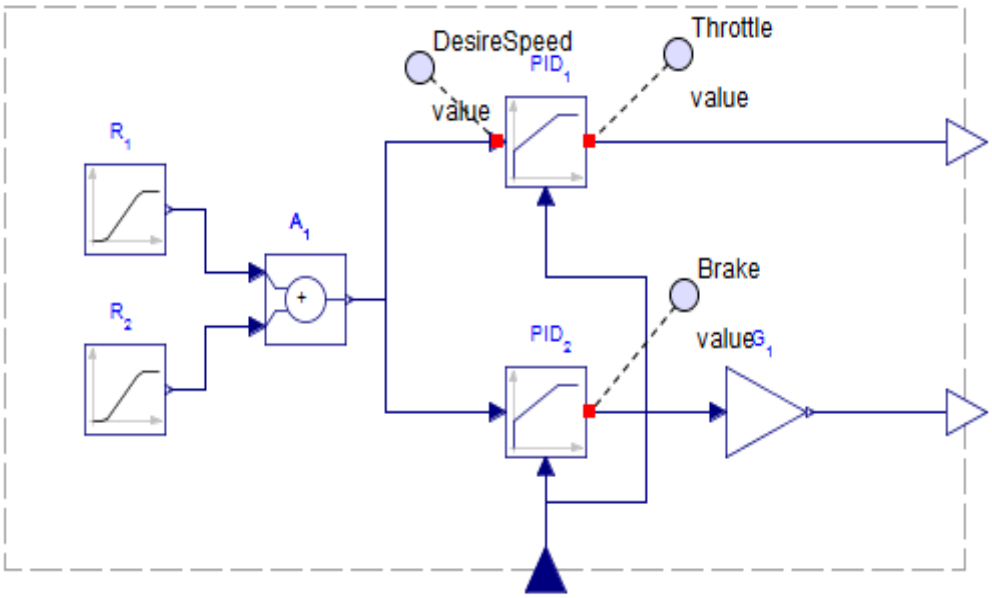


图 4-24　子系统添加探针位置

点击 Return to main（ ）按钮，回到主界面。

将测功机的速度作为反馈量，作为 PID Controller 子系统的输入量。并加入 Gain 元件。测功机的转速单位为 rad/s，设车轮半径为 0.35m。则设 Gain 元件变量的值为 3.6*0.35，使速度单位变为 km/h。按下图连接元件。

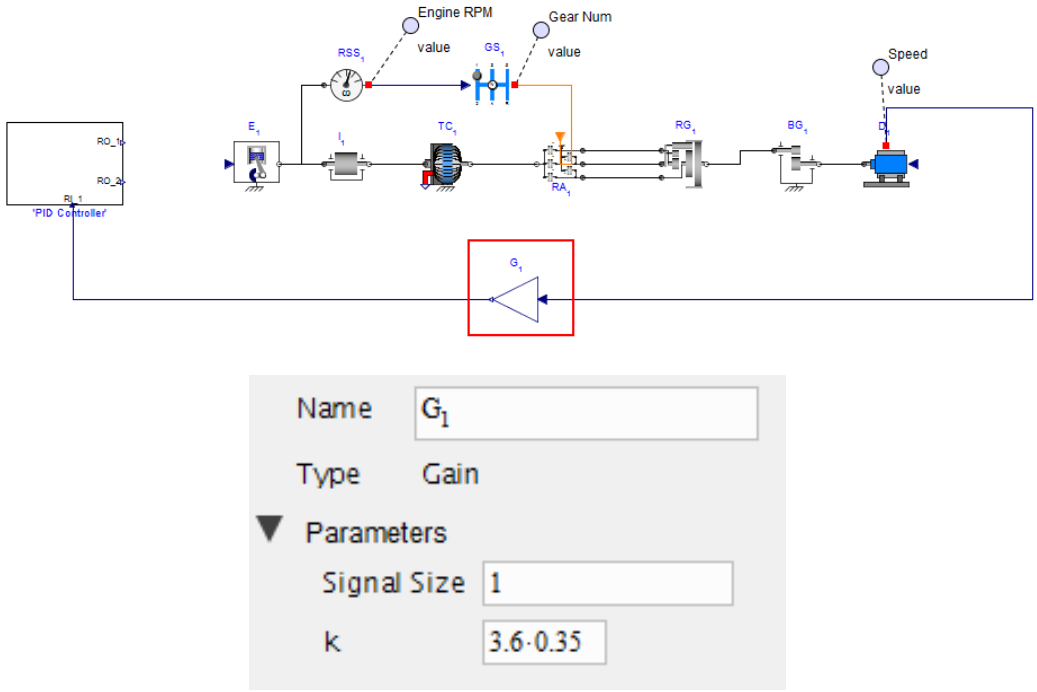


图 4-25　添加 Gain 元件并设定参数

将加速度信号与发动机连接，刹车信号与测功机相连。

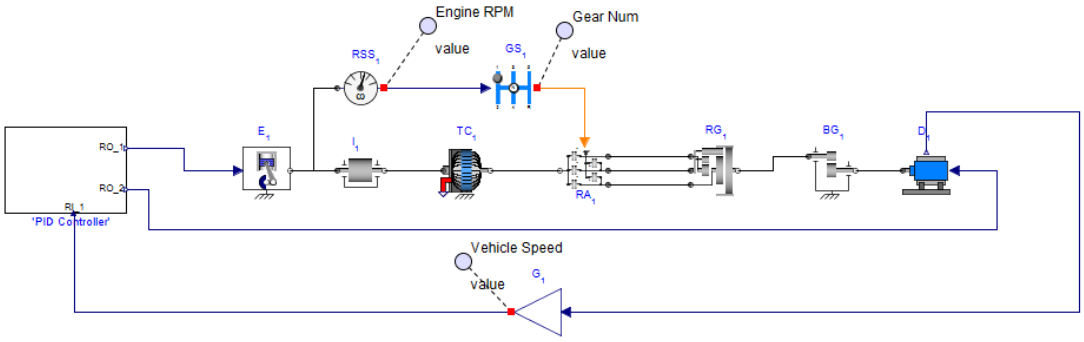


图 4-26　完整的系统模型

将仿真时间设定为 150s，并运行模型。

模型运行结果如下：

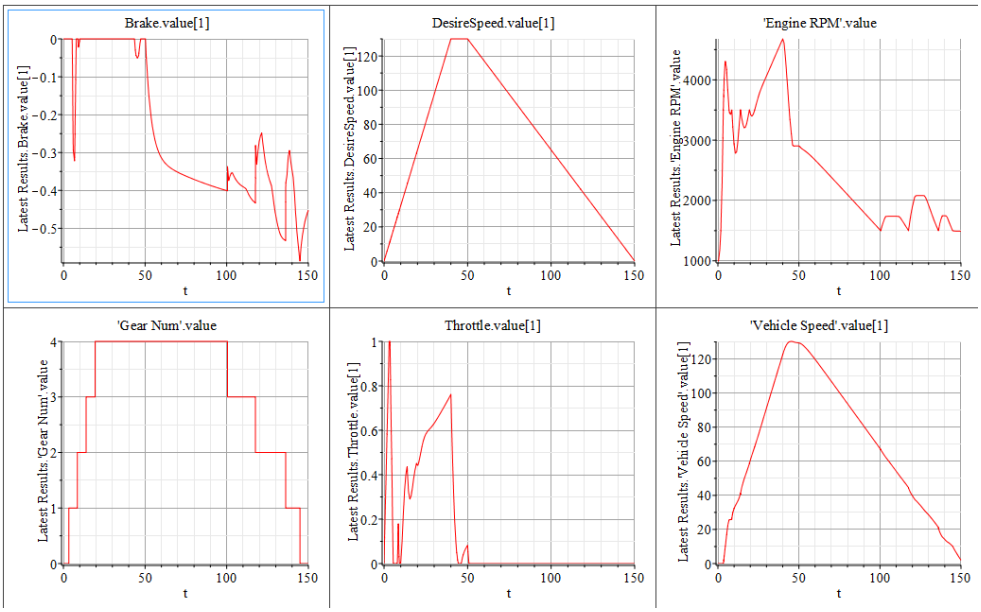


图 4-27　仿真结果图

选中 Vehicle Speed 曲线，并用鼠标拖入 DesireSpeed 窗口，得到理想速度曲线与实际速度曲线的对比图。

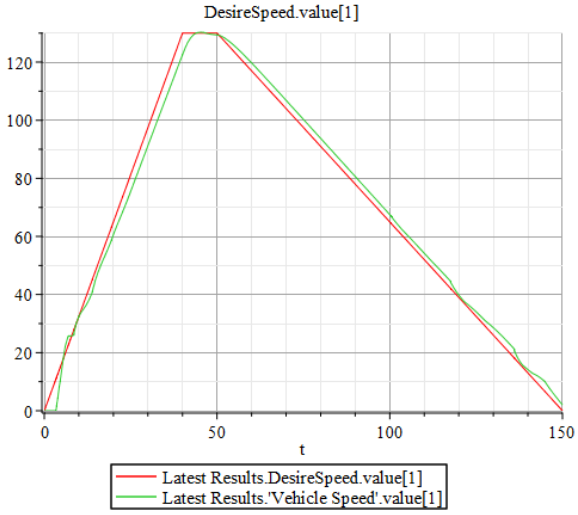


图 4-28　速度曲线对比图

第五章 RLC 电路和直流电机的建模仿真

概要

这个教程的目的是让用户熟悉 MapleSim 元件库和基础的模型开发工具。通过一些简单的示例演示如何混合因果模型和非因果模型。

在这个练习中，用户将完成下列任务：

1) 建立一个 RLC 电路模型
2) 设置元件参数定义仿真条件
3) 添加探针指定仿真结果中需要显示的值
4) RLC 电路模型仿真
5) 修改 RLC 电路创建一个简单的直流电机模型
6) 在不同参数条件下的直流电机模型仿真

模型下载地址：http://www.maplesoft.com.cn/book/5.1.zip

5.1 构建一个 RLC 电路模型

为了建立一个物理模型，用户需要添加元件到模型工作空间，然后连接它们组成系统。这个示例中，RLC 电路模型包括电子元件库中的接地、电阻器、电感、和信号电压元件。 它还含有一个阶跃输入信号源，这是一个信号发生器，控制电路中的电压大小。

1.在模型工作区左侧的 Labraries Components 选项卡中，点击 Electrical 旁边的三角标志展开面板。同样方式，展开 Analog 菜单，然后展开 Passive 子菜单，如下图所示。

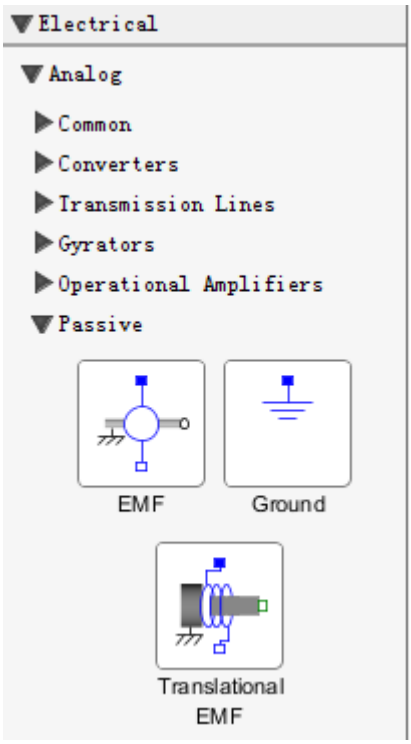


图 5-1　展开 Labraries Components 中的建模元件菜单

2. 从 Electrical → Analog → Passive 菜单，点击并拖动 Ground 图标到模型工作区，如下图所示。

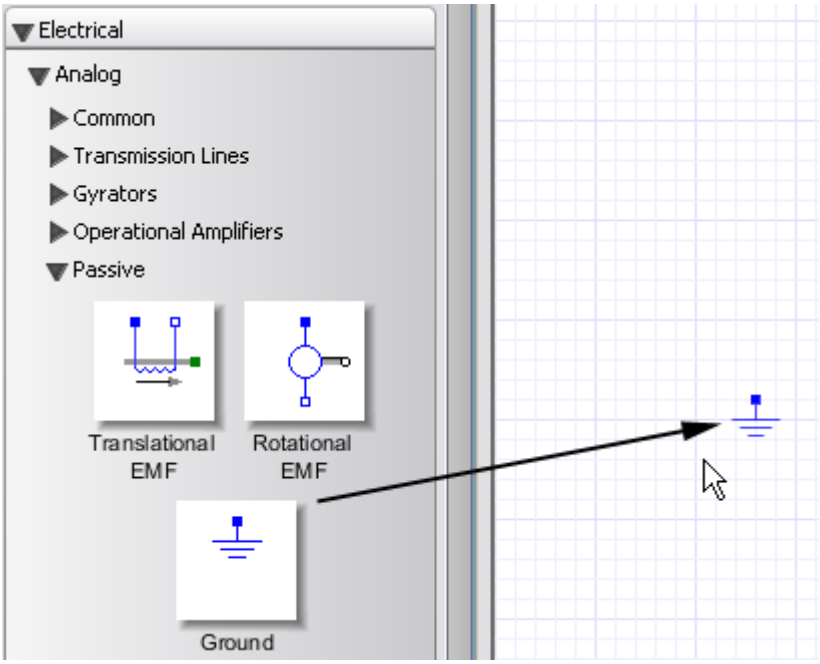


图 5-2　从建模元件库中拖入 Ground 元件到模型工作区中

3. 使用步骤 1 和 2 描述的方式，添加其余的电子元件到模型工作区：

- 从 Electrical → Analog → Passive→ Resistors 菜单，添加 Resistor 元件。
- 从 Electrical → Analog → Passive → Inductors 菜单，添加 Inductor 元件。
- 从 Electrical → Analog → Passive → Capacitors 菜单，添加 Capacitor 元件。
- 从 Electrical → Analog → Sources → Voltage 菜单，添加 Signal Voltage 元件。

4. 点击并拖动元件到如下图所示位置。

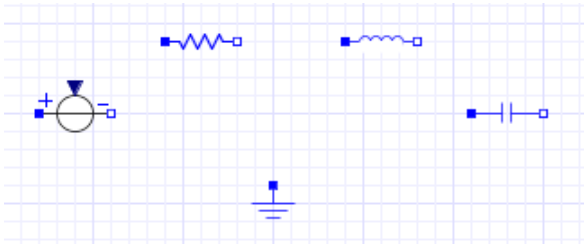


图 5-3　MapleSim 模型工作区中的元件布局

5. 顺时针方向旋转 Signal Voltage 元件：右键点击模型工作区中的 Signal Voltage 模块，从弹出菜单中选择顺时旋转(Rotate Clockwise)。

6. 水平翻转：再次右键点击该元件，从弹出菜单中选择水平翻转(Flip Horizontal)。确认正极（蓝色）端口在顶部。

7. 顺时针方向旋转 Capacitor 元件：右键点击模型工作区中的 Capacitor 模块并选择顺时针旋转(Rotate Clockwise)。

用户现在可以连接模型中的建模元件，定义系统内的相互作用。

8. 将鼠标指针移到 Ground 元件端口的上方，端口将高亮显示为绿色，如下图所示。

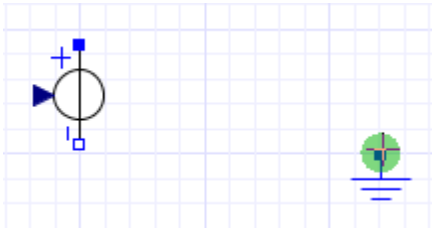


图 5-4　端口高亮显示的 Ground 元件

9. 单击 Ground 输入端口开始连接线。

10. 将鼠标指针移到 Signal Voltage 元件的负极端口，如图所示。

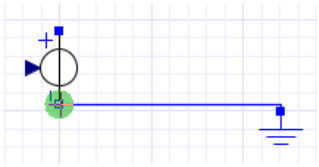


图 5-5　建立元件之间连接的示意图

11. 点击信号电压 Signal Voltage 输入端口。接地 Ground 元件被连接到信号电压 Signal Voltage 元件。

12. 按照下图所示连接其余元件。

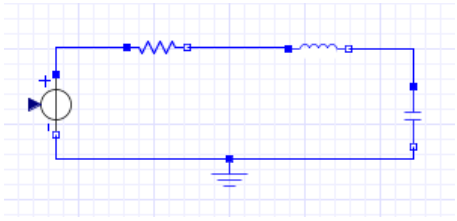


图 5-6　连接元件后的模型

用户现在可以添加一个信号源到物理模型中。

13. 展开信号块 Signal Blocks 面板。展开源 Sources 菜单，然后展开 Real 子菜单。

14. 从元件库中拖入 Step 源，放到信号电压 Signal Voltage 元件的左侧。Step 源是一个特殊的信号流，通过连线上的箭头表示方向。这个信号流产生电路对输入信号的响应。

15. 连接 Step 源到信号电压 Signal Voltage 元件。完整的 RLC 电路模型显示在下图 1-14 中。

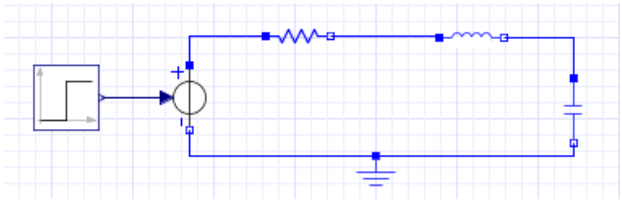


图 5-7　完整的 RLC 电路模型

5.2 定义元件的属性

在运行仿真前，用户定义模型中元件的参数值。

1. 点击模型工作区中的电阻器 Resistor 元件，参数面板出现在屏幕的右侧，显示电阻器的名称和参数值。

2. 在如下图所示的 R 区域内，输入 24，并按下回车键。电阻值改变为 24Ω。

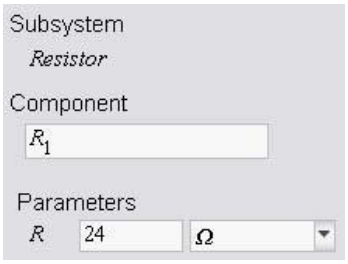


图 5-8　R 元件对应的电阻值定义面板

3. 使用步骤 1 和 2 的方式，定义其他元件的下列参数值：

- 定义 Inductor 的电感系数为$160\cdot10^{-3}$ H。
- 定义 Capacitor 的电容为$200\cdot10^{-6}$ F。
- 定义阶跃源 Step 中T_0的值为 0.1 s。

5.3 添加探针（Probe）

为了显示包含在仿真图形中物理量的值，用户需要在物理模型的连接线或端口上附加一个探针（Probe）。在这里，我们测量 RLC 电路的电压。

1. 从模型工作区工具栏上，点击探针按钮 ( )。

2. 移动用户的鼠标指针到 Inductor 和 Capacitor 之间的连线上，这条连线将高亮显示。

3. 单击连线。探针显示在模型工作区内。

4. 拖动探针到理想的位置。

5. 选中探针，探针的属性显示在模型工作区右侧。

6. 选择 Voltage 复选框，在仿真图形中显示电压量。

7. 为了让这个量在模型工作区中显示一个自定义名称，在下图所示的位置内输入 Voltage，然后按回车键。

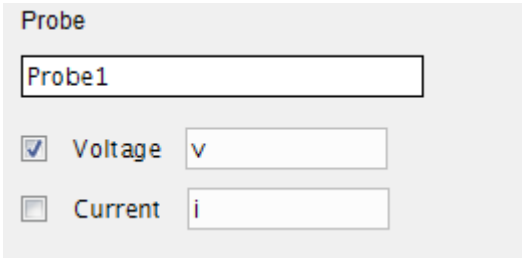


图 5-9　探针参数名定义面板

探针被添加到模块之间的连线上，如下图所示。

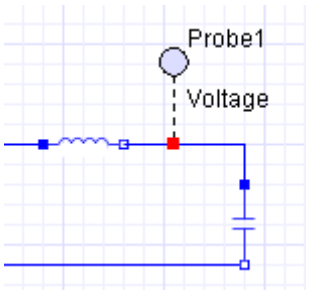


图 5-10　添加探针后的模型

5.4 RLC 电路模型仿真

在运行仿真前，需要定义仿真的运行时间周期。

1. 点击窗口右侧参数面板中的 Simulation Settings 选项卡，在 Solver 部分设置仿真结束时间 t_d 的值为 0.5 秒。

2. 设置 compiler 为 false。

3. 点击工具栏上的仿真按钮  。MapleSim 生成系统方程，然后仿真获得阶跃输入的响应。

仿真结束后，弹出电压响应图。

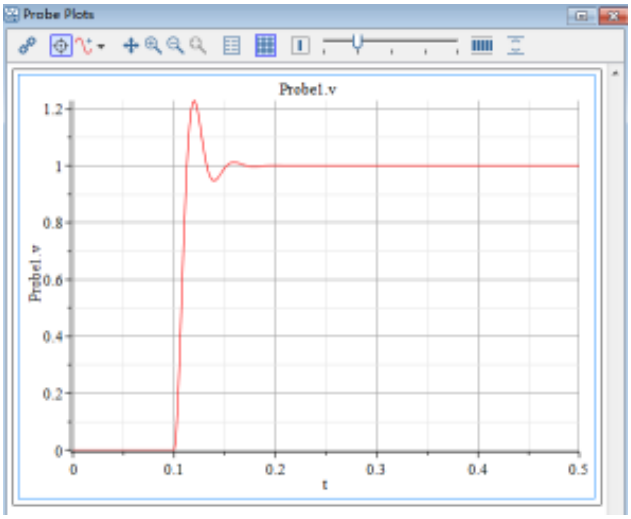


图 5-11 电压响应图

4. 使用文件名 RLC_Circuit1.msim 保存模型，探针和修改后的参数值被保存到模型中。

5.5 创建一个简单的直流电机模型

现在添加电动势 (EMF) 和一个机械惯量到 RLC 电路模型中创建一个简单的直流电机。在这个范例中，用户将使用搜索功能添加元件到 RLC 电路模型。

1. 在 Labraries Components 选项卡内，从 Electrical → Analog → Common 菜单，添加 EMF 元件。

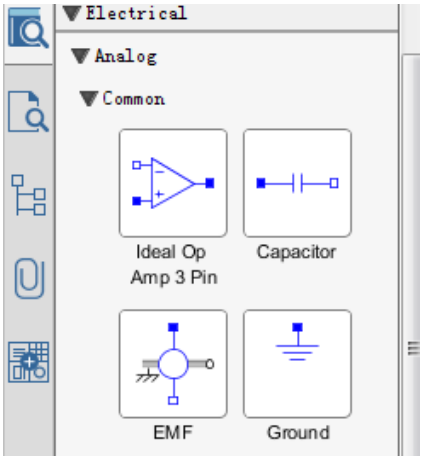


图 5-12 添加 EMF 元件

3. 从搜索区域中的方框区域，将 EMF 元件拖入到模型工作区，将它放在电容元件

（Capacitor）的右侧。

4.从 1-D Mechanical→ Rotational → Common → Capacitors 菜单，添加 Inertia 元件。

5. 添加惯量 Inertia 元件到模型工作区，放置在 Rotational EMF 元件的右侧。

6. 如下图所示连接各个元件。

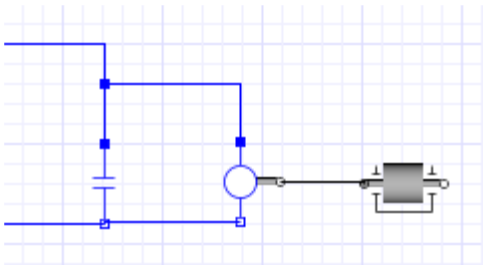


图 5-13　建立元件之间连接的模型示意图

为了连接电动势元件的正极（蓝色）端口，点击一下端口，拖动鼠标指针到电容和电感之间的连线，并点击进行连接。

7. 在模型工作区，点击 EMF 元件。

8. 在右侧 Properties 选项卡，修改转换系数的值 k 为$10\frac{N\cdot m}{A}$。

9. 点击阶跃元件 Step，修改参数值 T_0 为 1。

5.6 直流电机模型仿真

1. 在模型工作区，删除探针 Probe1。

2. 在模型工作区工具栏上，点击探针按钮（  ）。

3. 移动用户的鼠标指针到 Rotational EMF 和 Inertia 元件之间的连线上方。

4. 鼠标左键点击一下连线，然后再点击一下，定位探针的位置。

5. 选择探针。

6. 在 Properties 选项卡内，选择速度 Speed 和扭矩 Torque 复选框。探针添加到模型中，箭头表示方向。

7. 点击模型工作空间的空白区域。

8. 在 Settings 选项卡内，设置仿真结束时间 t_d 为 5 秒，然后点击主工具栏上的运行仿真按钮 。显示下图所示的仿真结果图。

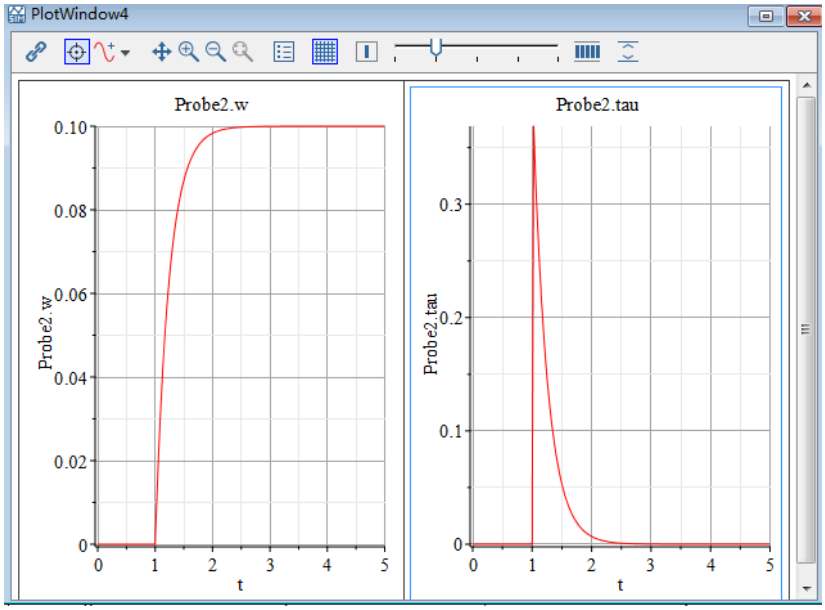


图 5-14　MapleSim 仿真结果图

9. 使用文件名 DC_Motor1.msim 保存模型，以便后面的章节使用。

第六章 车辆热系统建模

6.1 车辆热系统建模

模型下载地址：http://www.maplesoft.com.cn/book/6.1.zip

打开文件“HEV_Powertrain_Cooling_FTP_Cycle.msim”文件。

这个模型代表一个混合动力车冷却系统模型。模型是基于物理锂离子电池组,发动机,电机/发电机,功率控制器和冷却系统组成的。冷却液被冷却液泵推进电池和发动机模块，热量被散热器带入周围空气中。通过检测风扇和车速来控制散热器空气流动。

点开子系统 Cooling，打开散热器子系统。

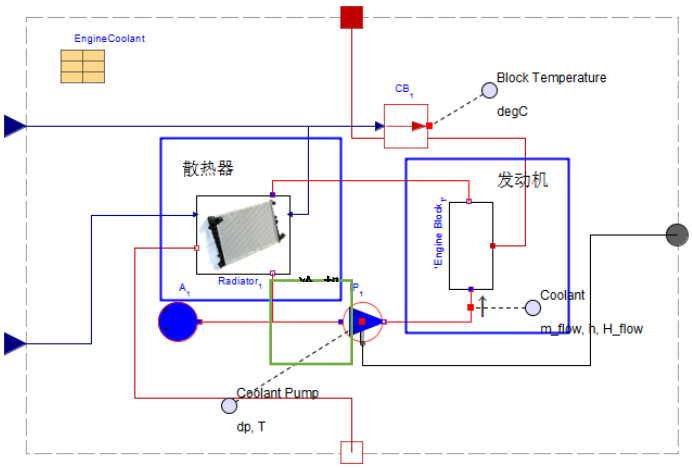


图 6-1 Cooling 子系统

1、添加加热管元件；Thermal > Pipes and Valves > Heated Pipe

m=Mfluid_rad kg ；勾选 Fixed T0 ;

添加热容元件；Thermal > Heat Transfer Components > HeatCapcitor

C=896.3 J/K;

添加热传导元件；Thermal > Heat Transfer Components > Thermal Conductor

$$G=\frac{220*2Pi}{\log\left(\frac{3}{2}\right)}\bullet 5\frac{W}{k};$$

2、添加热对流元件；Thermal > Heat Transfer Components > Convection; 热传导率由实信号决定。

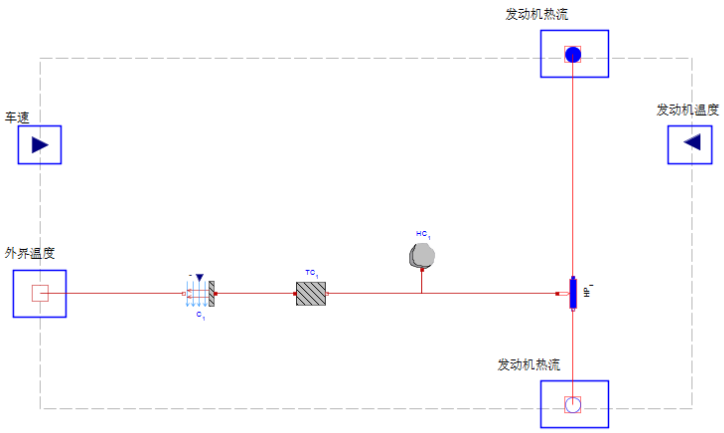


图 6-2　添加元件后的模型

3、单位转换元件；Signal Blocks > Signal Converters > Conversion Block

4、信号处理元件；Signal Blocks > Mathematical > Operators > Product

5、参数表 ；　Signal Blocks > Interpolation Tables > Look up Table 1D

data = RadiatorHeatTransfer.xlsx　　column=[2]　　skiprows=1

RadiatorHeatTransfer.xlsx 表格是用查表的方法，查找相应车速下定义 Convection 中空气与散热器的对流热导率。

6、Signal Blocks > Boolean > On Off Controller

hysteresis=5 ；　勾选 pre_{y0} ；

Signal Blocks > Signal Converters > Boolean To Real

Real True=4.92 ; Real False=0 ;

7、2 个单位转换元件

Signal Blocks > Signal Converters > Conversion Block

CB1:　dimension=Velocity;　fromUnit=km/h;　toUnit=m/s;

CB2:　dimension=Temperature;　fromUnit=^{o}C ;　toUnit=K;

8、信号源元件

Signal Blocks > Common > Constant

k=SetTemp;

9、按下图所示连接元件

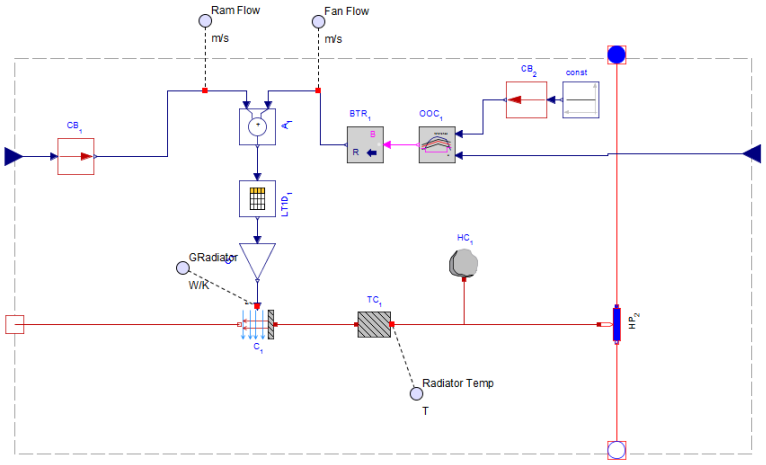


图 6-3 完整的热系统模型

10、运行模型，查看温度，冷却剂，热流等。

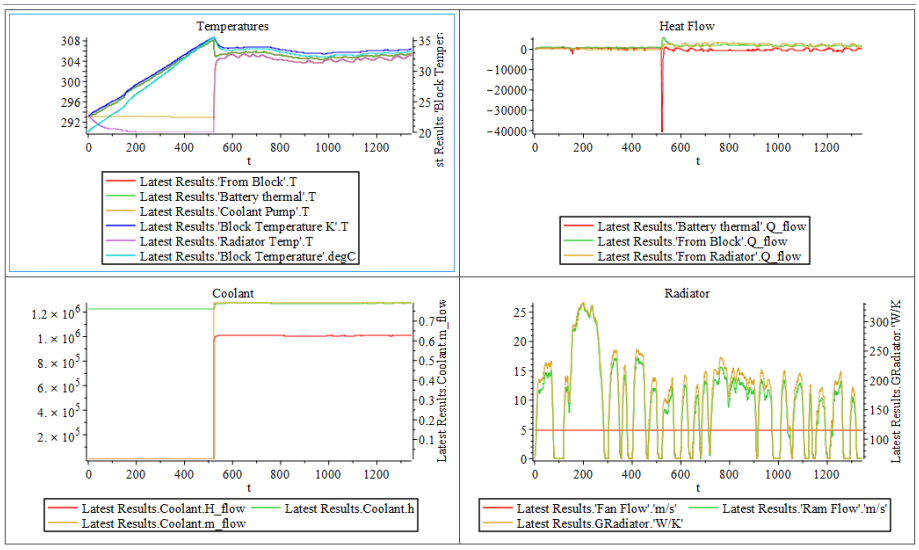


图 6-4 仿真结果图

第七章 硬件在环模型设置和实现

7.1 项目背景

汽车作为一个高度复杂的产品，涉及到机械，电子，液压，控制，化学，热等多个物理领域，同时对安全的高标准和成本的控制，使得汽车的设计成为一个巨大挑战，同时由于较长的开发周期，在开发早期对整个系统的功能和性能进行评估对于保证开发质量和时间具有重要意义。传统的研究方法需要用到很多传感器、测试设备和数据采集装置和专用的试验场地，通过大量的实车试验，来获得所需要的数据和信息。同时，对实车进行极限条件下的试验有很大的风险，这无疑造成研发周期的延长，效率的降低，成本的提高等诸多弊端。基于模型的开发和测试已被汽车行业证明是高效和可靠的方法。虚拟驾驶实验室通过建立车辆的仿真模型及道路环境模型，进行人机交互与实时仿真，对车辆行驶的性能进行评估与分析，可缩短研发周期、降低成本、提高效率，有效的解决了实车实验所带来的不便。

理论研究方面，采用模拟驾驶系统可以根据实际需要改变仿真车辆的结构参数和仿真条件，而且可以记录下车辆行驶过程中各个部分的状态参数的变化，以多种形式（图形、动画、数据、曲线）表现出来，对于分析车辆的动力学性能、研究驾驶员的操纵行为都有很重要的意义。这样，整车模型经过参数的标定，可以在虚拟的环境下完成车辆的各种实验和测试，相对于实际的车辆原型，使用物理模型的方法具有如下优势:

- 可以在设计的早期进行实验和验证；
- 测试环境易于控制，特别是具有危险性的故障模拟；
- 计算机模拟的速度可以控制，测试时间大大缩短；
- 可以实现复用，参数或某些部件修改后即可模拟其他车辆的行为和特性。

实际应用方面，应用该系统进行汽车动力学仿真，可以为道路设计和驾驶员行为引起的车辆操纵失稳的研究提供实验平台。对于交通安全方面，根据交通事故现场的勘测及车辆的相关数据，可对某一事故多发路段进行建模仿真，通过建立这一路段的行驶环境，重现事故发生时车辆的行驶状态，从而找出导致事故发生的主要原因和保持车辆安全稳定驾驶的方法，指导驾驶员如何在此路段操纵车辆避免事故的发生；该系统还可以作为学习车辆动力学相关知识的实验平台。

7.2 系统构成

虚拟驾驶平台建设需求搭建硬件在环（HIL）系统。虚拟驾驶硬件在环系统是由MapleSim 车辆模型、视景系统，Concurrent 实时机以及实体的方向盘、踏板组成的半实物仿真系统。

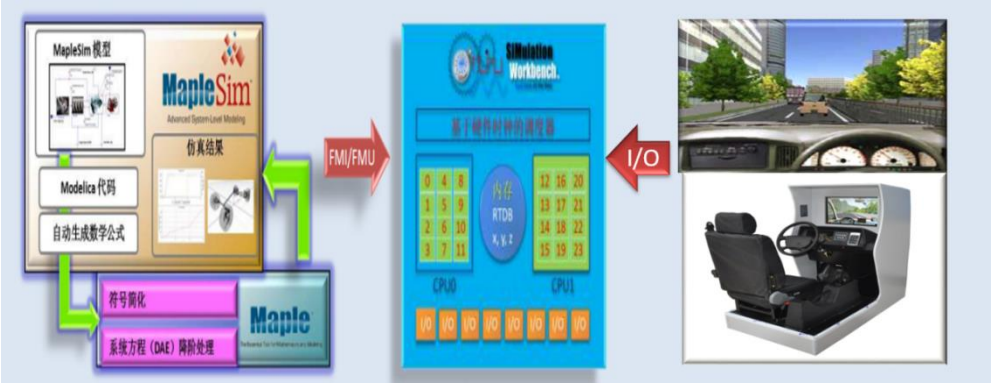


图 7-1　虚拟驾驶硬件在环系统

7.3 FMI 接口

为了实现一体化仿真平台系统的通用性，采用了 FMI(Functiona Mock-up Interface)标准化接口。FMI 的目标是定义一个开放的接口，从而让不同软件系统的动态系统模型能够实现软件模型硬件的联合仿真。因此，FMI 接口对于现有的主流商业仿真软件和程序语言有良好的兼容性。

FMI 标准接口包含模型描述文件和仿真程序代码两个部分。模型描述文件为 XML 格式，主要用于描述模型属性，包括模型名称输入输出参数等。仿真程序代码为 DLL 文件，主要是用 C、C++或是 MATLAB 编写的源程序。

DLL 文件内部要求按照标准接口函数编写，仿真软件将上述 DLL 模型实例化后生成仿真单元 FMU，仿真软件按照标准调用方式对 FMU 进行调用完成仿真。仿真软件需要为 FMU 提供外部求解器。

MapleSim 提供了 FMI 接口工具箱，并通过 FMU 文件与 Concurrent 的实时平台进行连接。

7.4 MalpeSim 模型

1、MapleSim 模型介绍

模型下载地址：http://www.maplesoft.com.cn/book/7.1.zip

MapleSim 车辆模型是由车身与传动系统构成，如图所示。

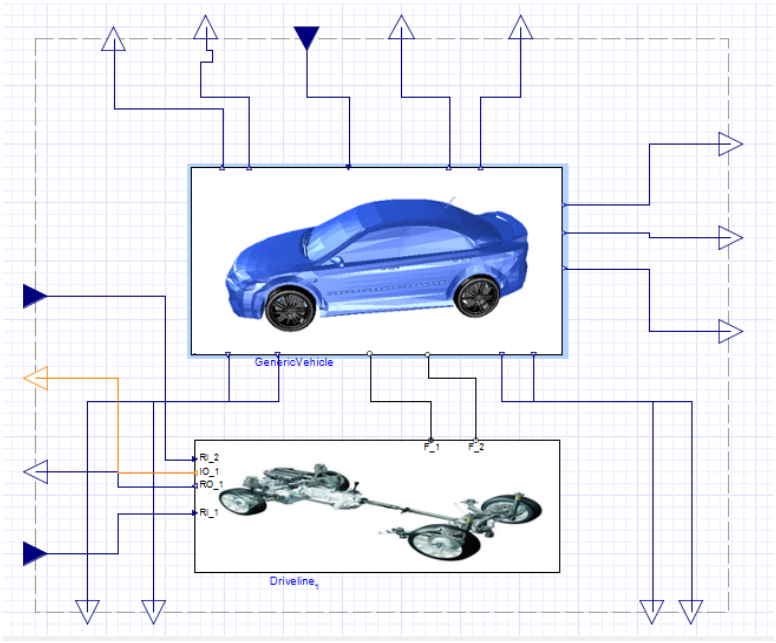


图 7-2 MapleSim 车辆模型

通过硬件设备方向盘，油门，刹车控制车辆的运行，我们将这三个变量设为输入量。为了与视景系统交互，需要输出车身的位置与姿态，车轮的位置与姿态，车辆行驶速度等。

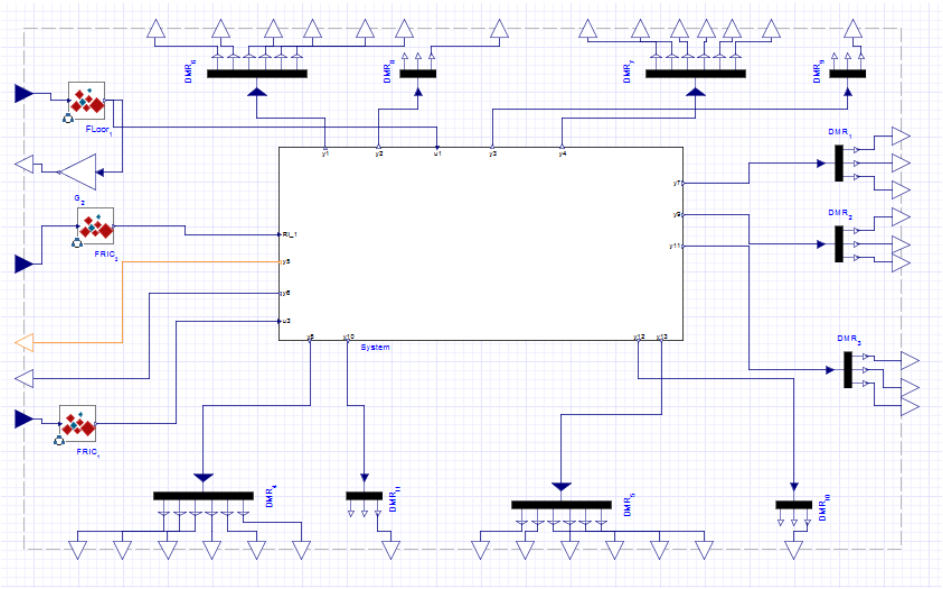


图 7-3 MapleSim 输入输出接口

2、模型求解超时问题

（1）模型求解时间过长

例：外接设备输入变量数值精度过高

由于外接设备精度较高为小数点后六位，模型求解的过程较长，可能出现超时的情况。我们利用自定义元件 FRIC 降低输入变量的精度至小数点后四位。

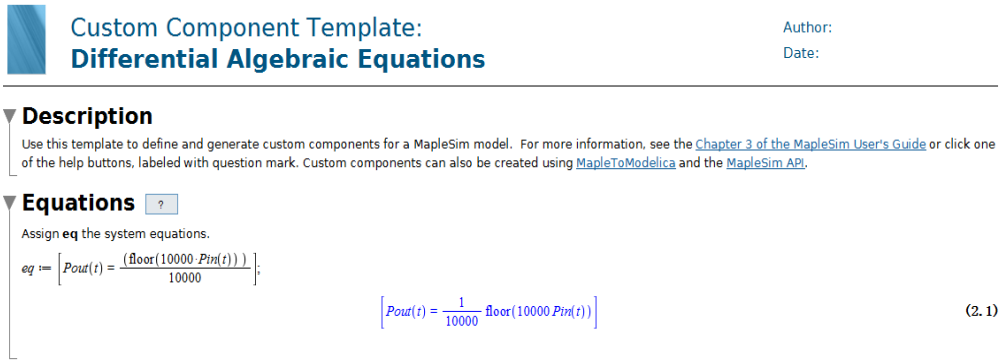


图 7-4 FRIC 自定义元件

(2) 模型求解发散

模型发散的原因：模型求解不连续，变步长元件影响过大。

解决方法：确定变步长元件，调整模型使模型求解连续。

(3) 仿真步长的确定

仿真步长过大，模型求解发散的可能性增大。过小的仿真步长，可能导致模型在一个仿真步长的时间很难求解结束。所以需要综合两种情况，考虑到模型的大小与求解发散的可能，我们虚拟驾驶模型选择 1ms 的仿真步长。

7.5 生成 FMU 文件

在 MapleSim 中打开相应的模型

- 点击 Add Apps and Templates。
  - 在对话框中双击 FMU Generation。

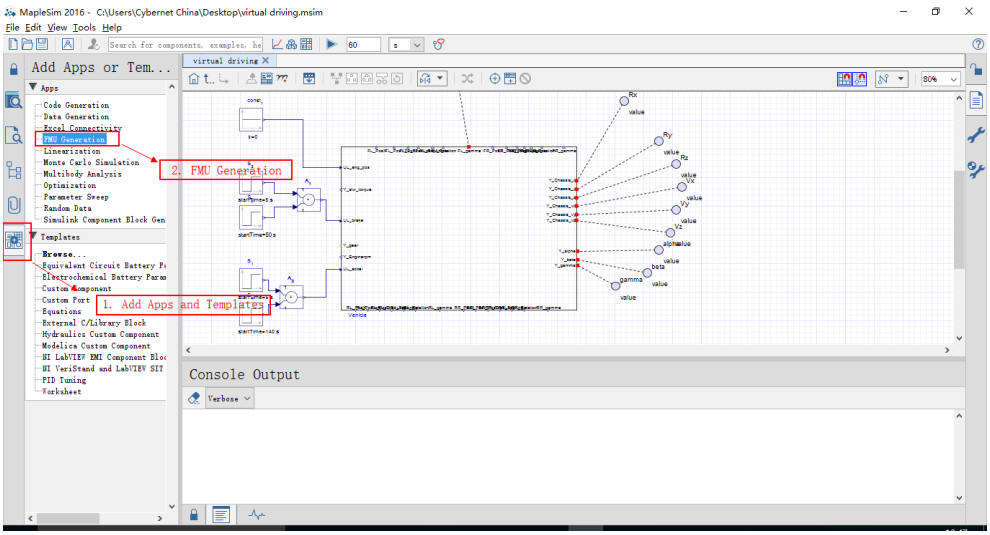


图 7-5　FMU Generation

- 后弹出 FMI Component Code Generation，选择需要输出 FMU 的子系统。
  - 点击“load selected subsystem”。

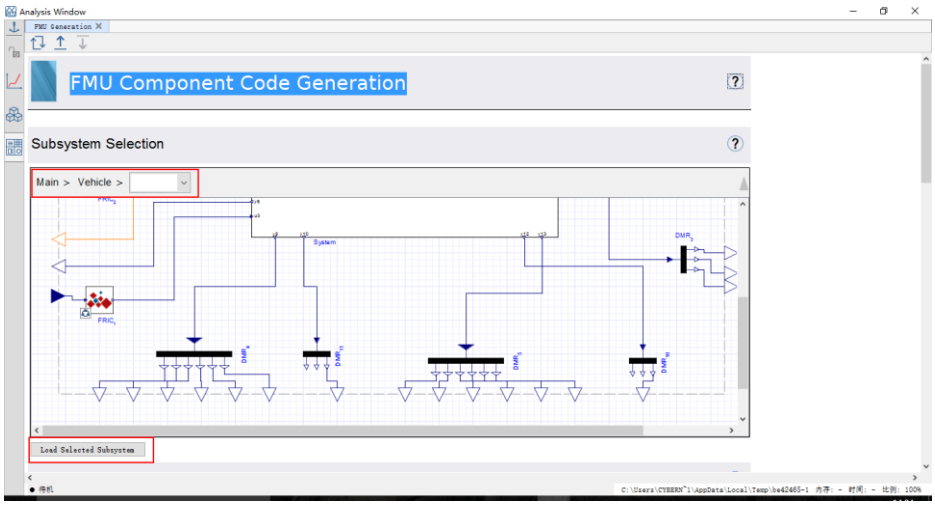


图 7-6　载入模型

- 分别点击 Input, Output, Parameters 查看模型输入变量、输出变量以及参数。

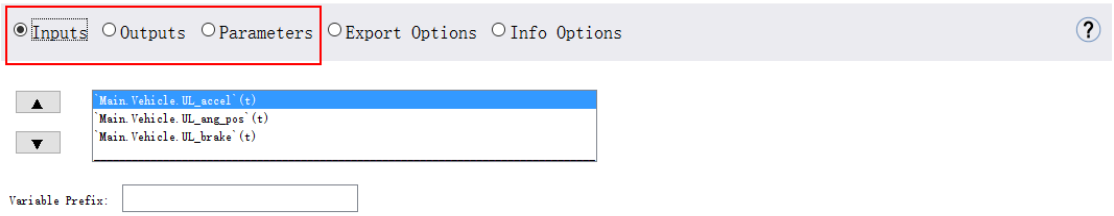


图 7-7　定义输入输出变量

- 点击 Export Options，设定输出选项。

注：最好每个参数与 MapleSim 仿真设定（Simulation Setting）相同，从而保证生成代码与 MapleSim 模型的一致性。

（1）约束处理选项

Constraint Handling Options 定义约束是否满足使用约束投影生成 Simulink 块的微分代数方程 DAE 系统。使用这个选项提高包含约束的 DAE 系统的精度。如果约束没有满足，系统结果会偏移实际的结果，并导致错误呈指数增加。

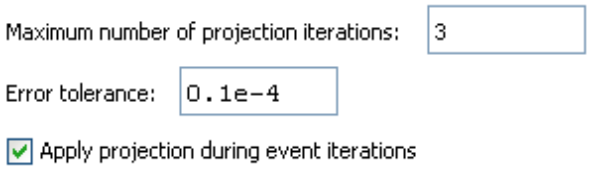


图 7-8　约束处理选项设置

设置 Maximum number of projection iterations 定义投影允许的最大迭代数，获得更精确的结果。

设置 Error tolerance 定义期望的投影误差容许。

选择 Apply projection during event iterations 使用插值迭代获得更精确的结果。

约束投影（Constraint projection）使用 External Model Interface 中的 constraint projection 程序，控制 DAE 系统的偏移，详细信息见 Mathworks 的网站。

（2）事件处理选项

Event Handling Options 定义约束是否满足使用事件投影生成 Simulink 块的微分代数方程 DAE 系统。使用这个选项提高包含事件的 DAE 系统的精度。如果事件没有满足，系统结果会偏移实际的结果，并导致错误呈指数增加。

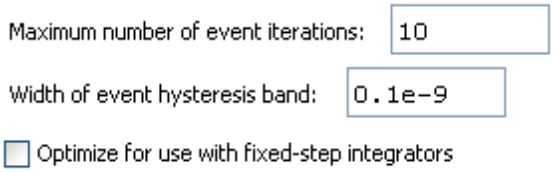


图 7-9　事件处理选项设置

设置 Maximum number of event iterations 定义投影允许的最大迭代数，获得更精确

的结果。

设置 Width of event hysterias band 定义期望的投影误差容许。

选择 Optimize for use with fixed-step integrators 使用 hysterias bandwidth 的函数优化事件迭代。

事件投影（Event projection）使用 External Model Interface 中的 event projection 程序，控制 DAE 系统的偏移，详细信息见 Mathworks 的网站。

（3）定步长积分选项

设定求解器与仿真步长。（最好与 MapleSim 中仿真求解器与仿真步长一致，避免实时机对的仿真结果与 MapleSim 仿真结果不一致）。

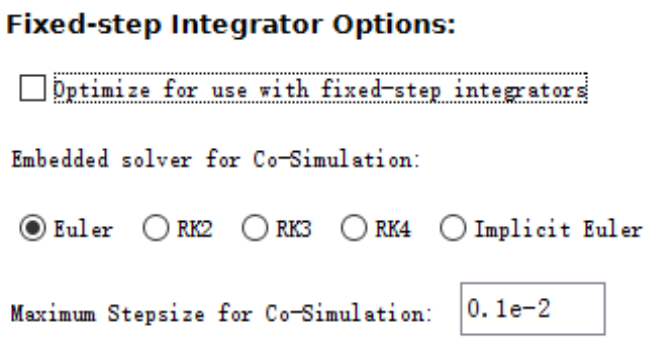


图 7-10　定步长积分选项

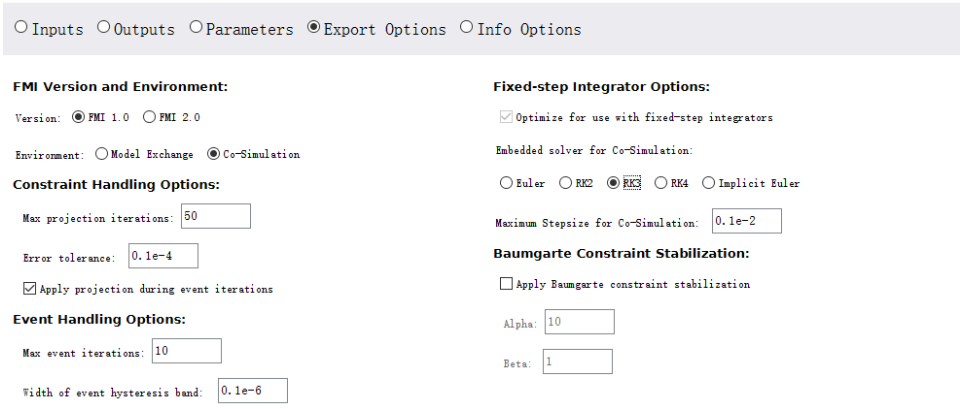


图 7-11　模型导出选项

- 选择保存路径，生成文件名，最后点击 Generate FMU Archive。这样就在对应的路径下生成了我们需要的 FMU 文件。

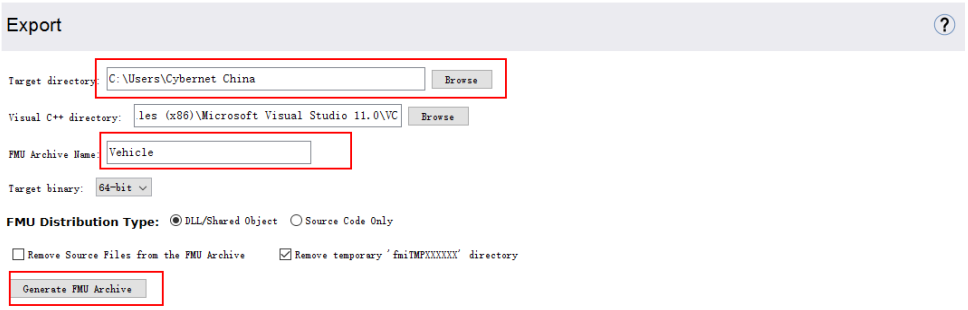


图 7-12　生成 FMU 文件

7.6 在 Simulation Workbench 中导入 Maplesim 模型

在 SimWB 中导入流程如下：

- 在主控界面上下拉菜单“Third Party Tools”选择 FMU。

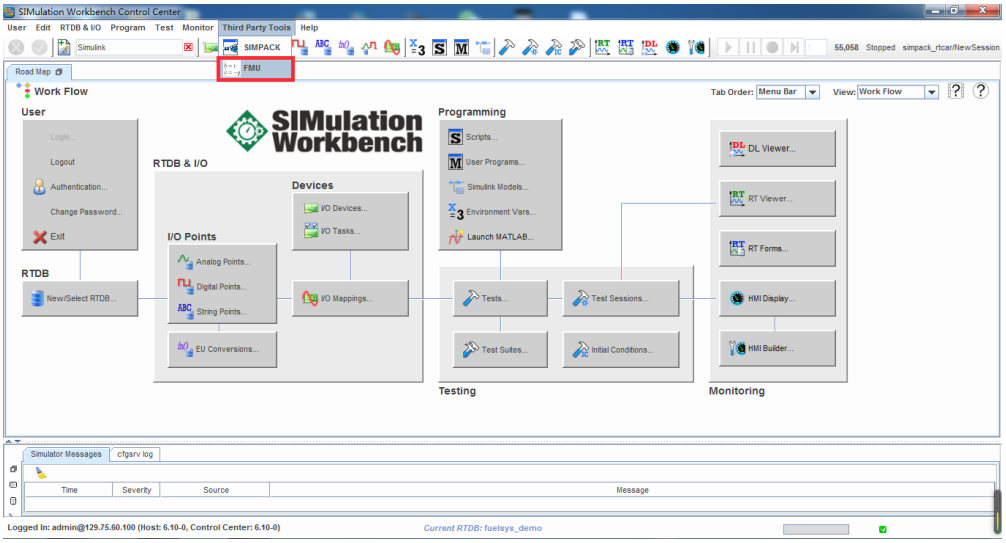


图 7-13　导入 FMU 文件

- 之后在打开的 FMU 导入界面选择相应的 FMU 文件。

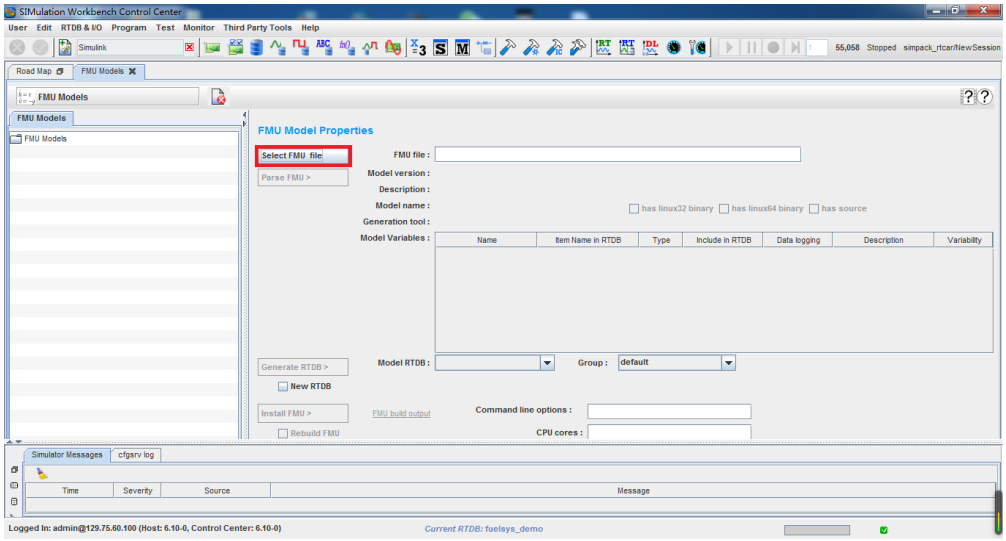


图 7-14　Select FMU Files

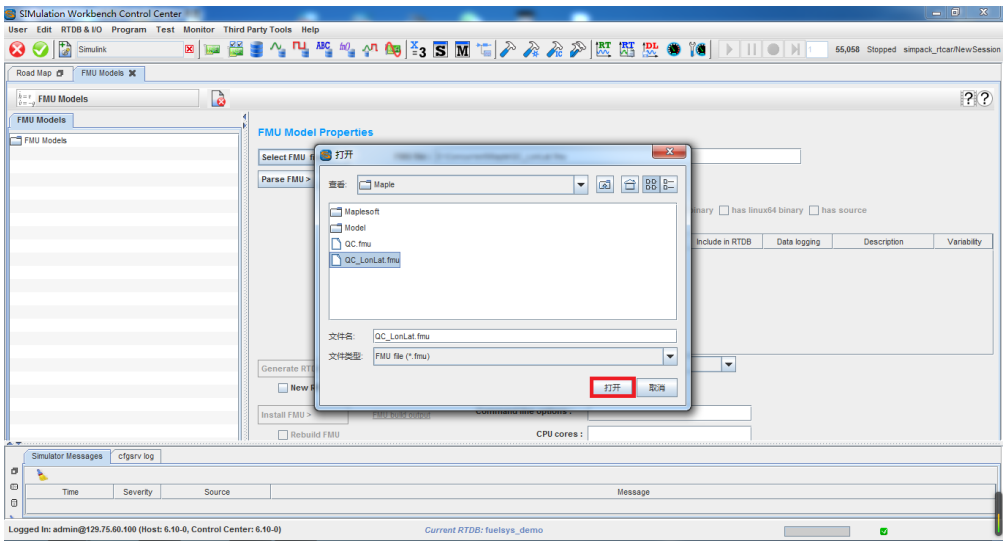


图 7-15　打开所选 FMU 文件

- Parse FMU 文件，相关的信息可以显示，包括版本号，生成工具，以及模型中的输入输出变量，默认情况下输入输出变量是会生成 RTDB 变量的。

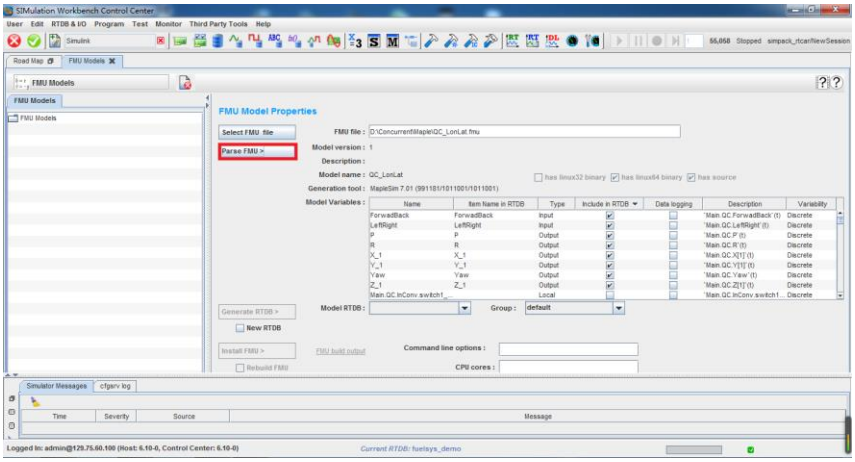


图 7-16　Parse FMU

- 生成 RTDB 数据库，模型中所有的输入输出变量自动生成 RTDB 变量。

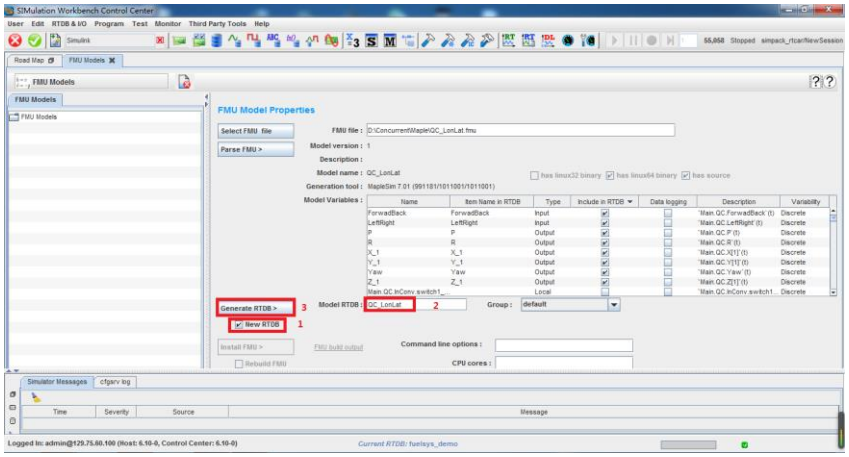


图 7-17　生成 RTDB 数据库

- 设置模型运行的 CPU。

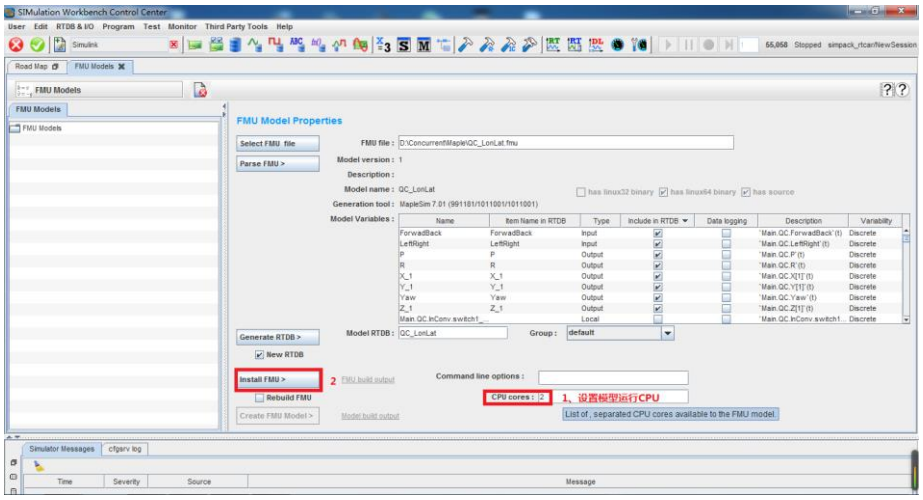


图 7-18　设置模型运行 CPU

- 生成模型的可执行程序，这样一个完整 Maplesim 的 FMU 模型就导入到 SimWB 中，并且可以运行。

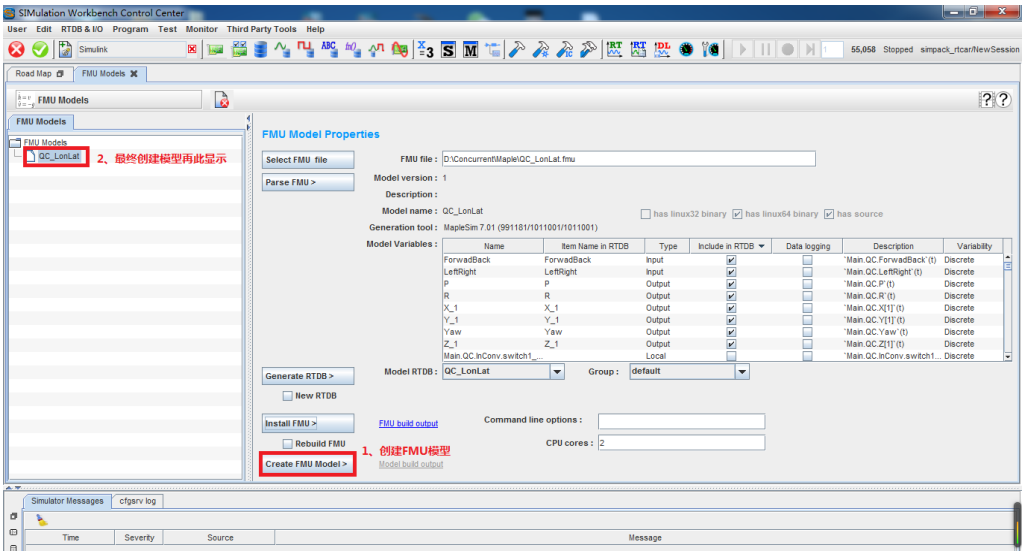
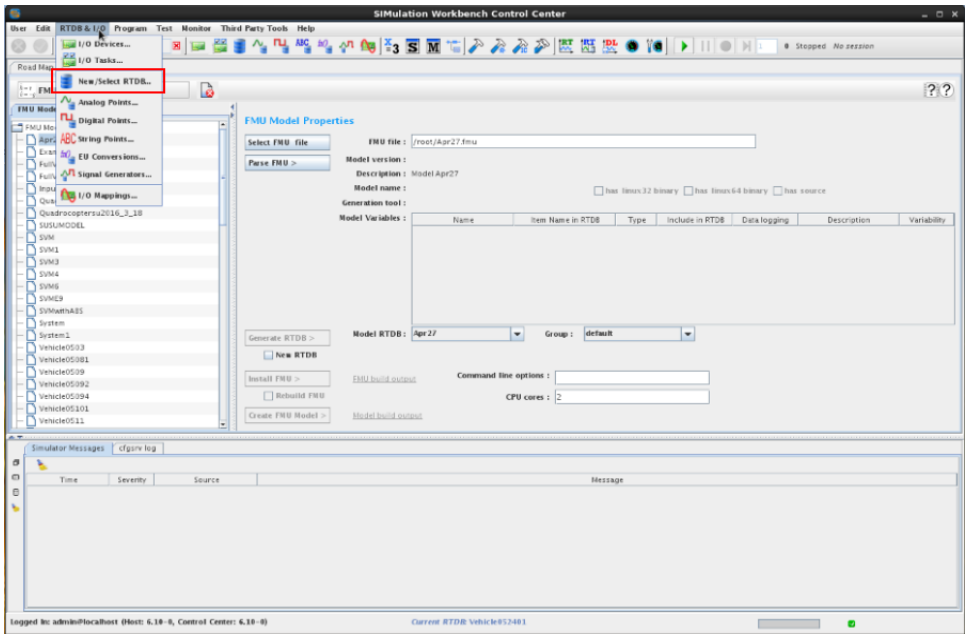


图 7-19　生成 RTDB 执行程序

- 点击 New/Select RTDB 查看新生成的 RTDB。点击刷新，可以在列表中找到新生成的 RTDB 模型。



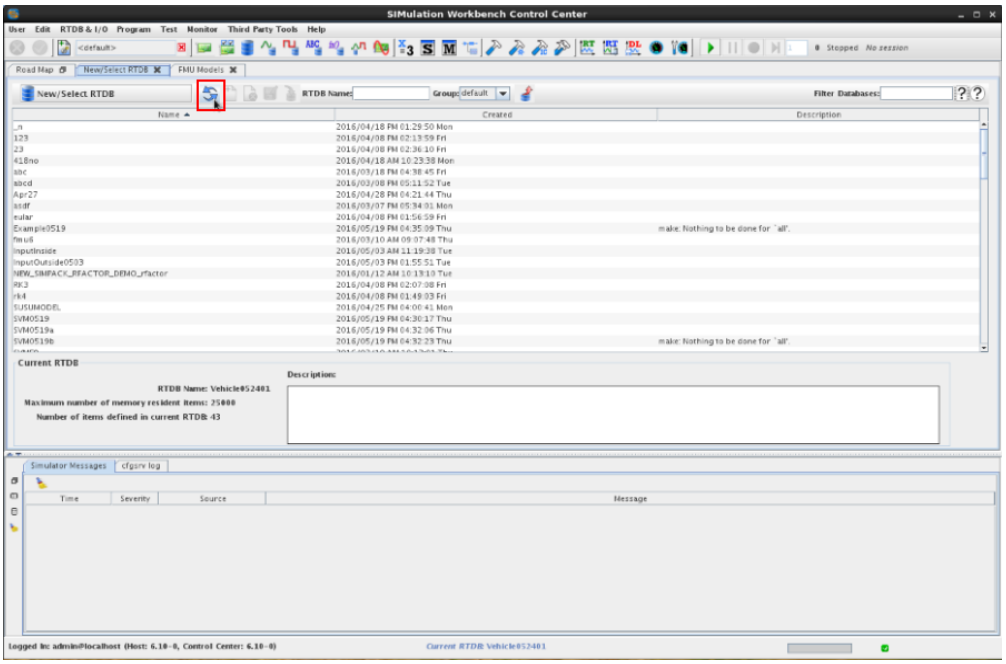


图 7-20　查看新生成的 RTDB

- 点击 I/O Devices，导入硬件设备，这里，我们选中罗技方向盘为硬件设备，并添加到设备列表。

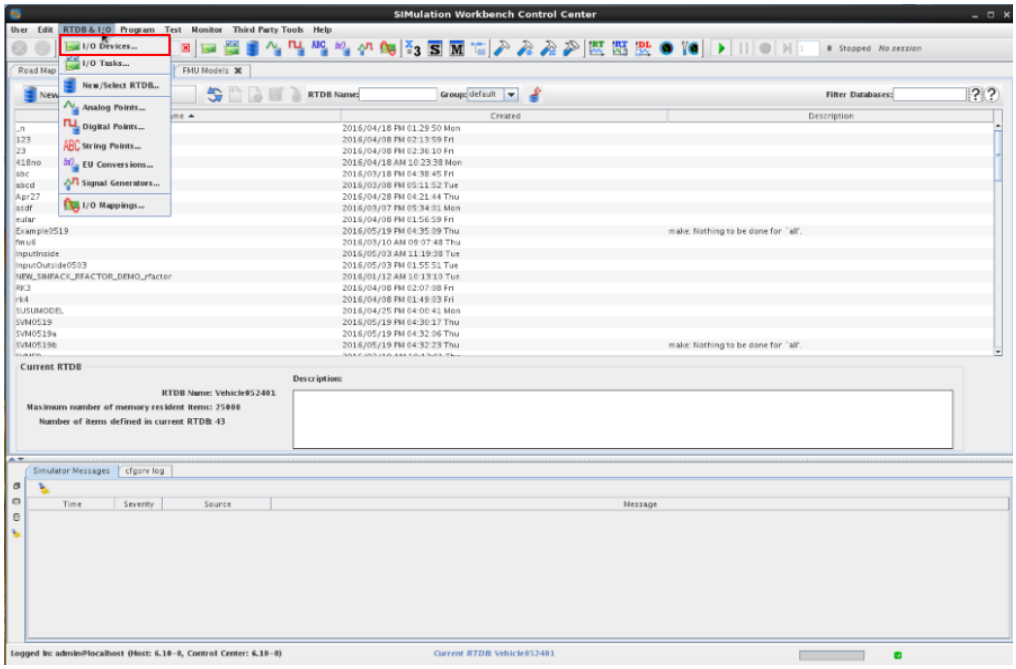


图 7-21　导入硬件设备

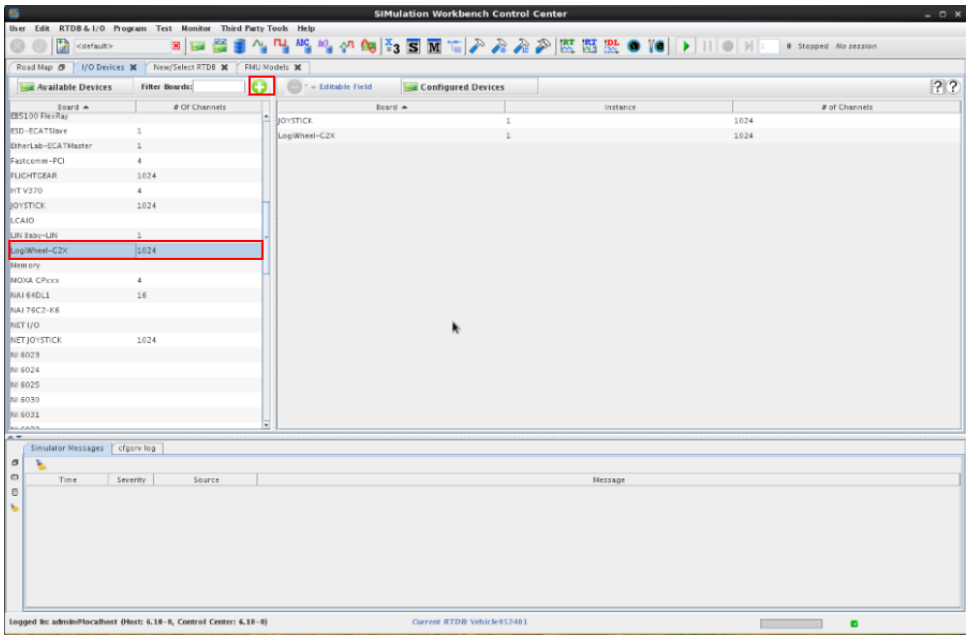


图 7-22　添加硬件设备

- 点击 I/O Mappings 使硬件设备与模型的输入输出量对应。

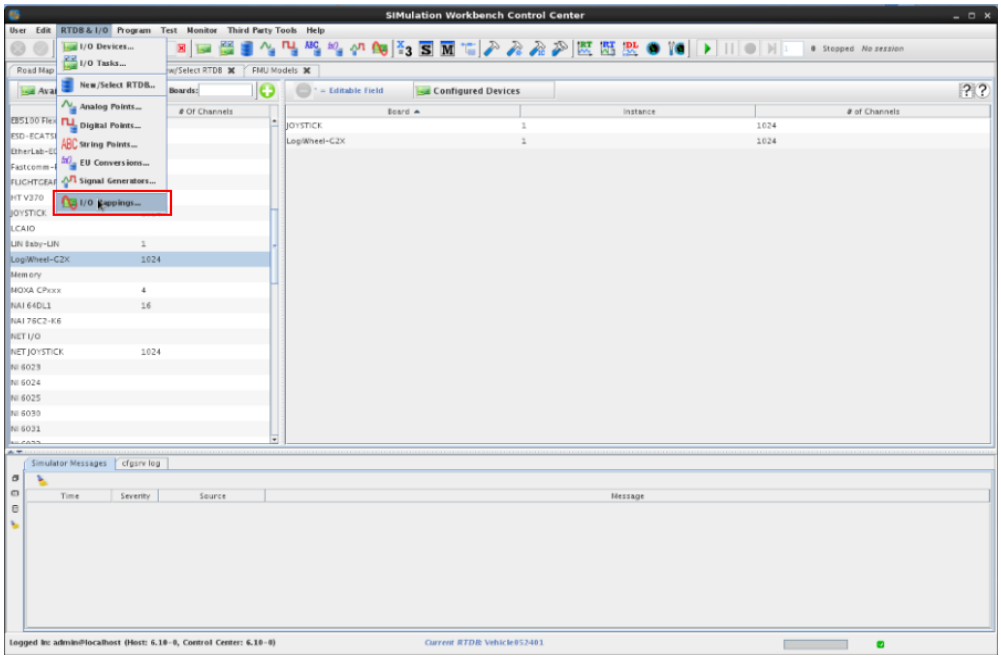


图 7-23　I/O Mappings

选中硬件输出变量后，在右侧列表中对应的模型输出变量前打勾。例如图中的方向盘反馈力，并选中右侧模型输出的方向盘反馈力的数值。

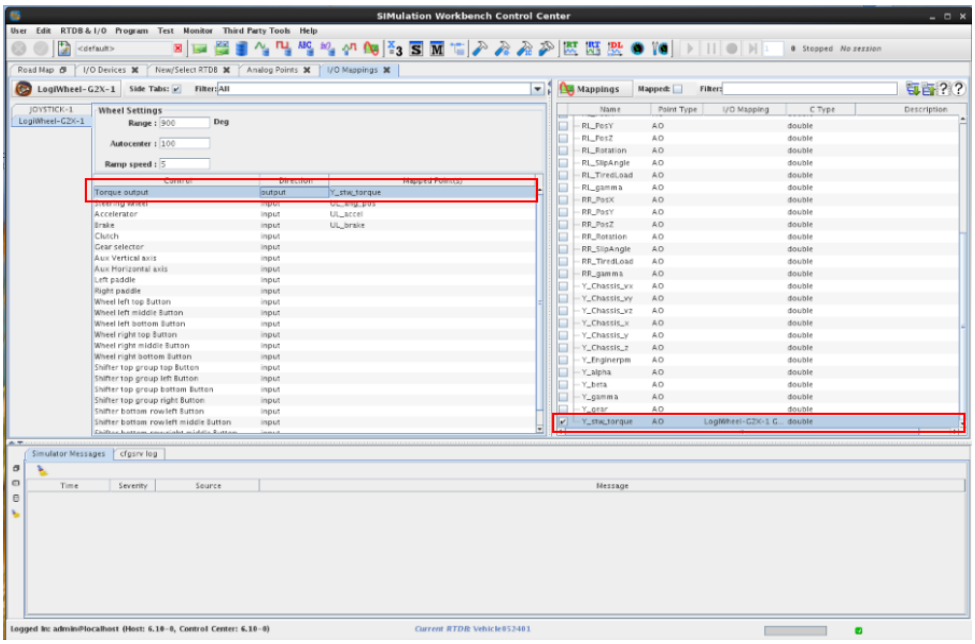


图 7-24　方向盘力反馈与模型输出变量对应

- 点击 Analog Points 定义模拟输入输出点的值。

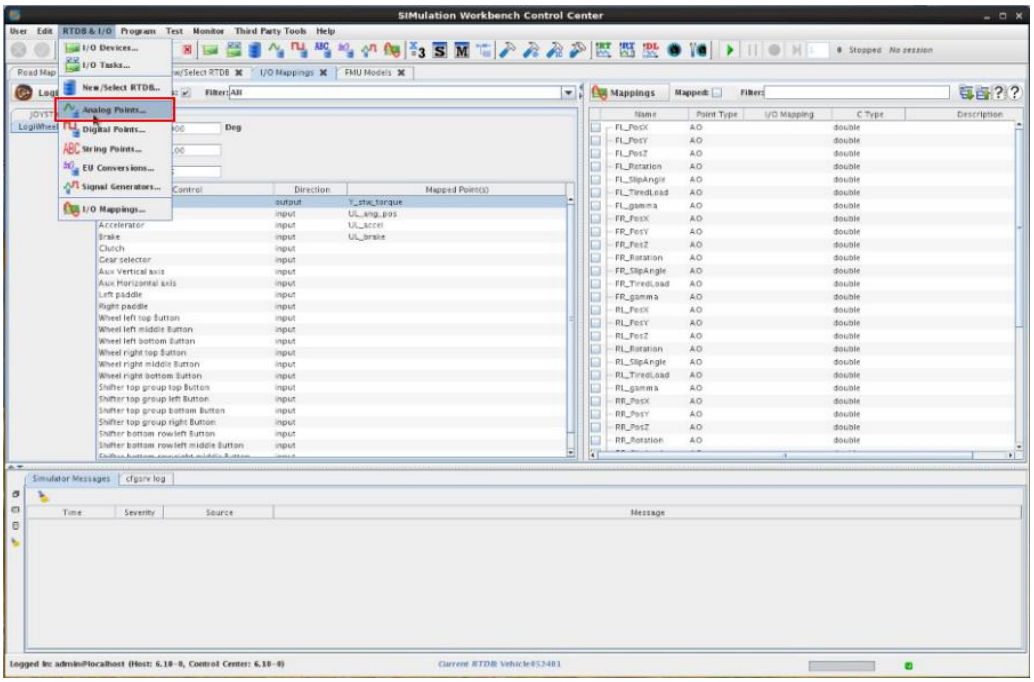


图 7-25　Analog Points

例如，将方向盘的转角实际角度缩小十倍，输入到模型中。选中表示方向盘转角的变量，编辑变量转换的值。

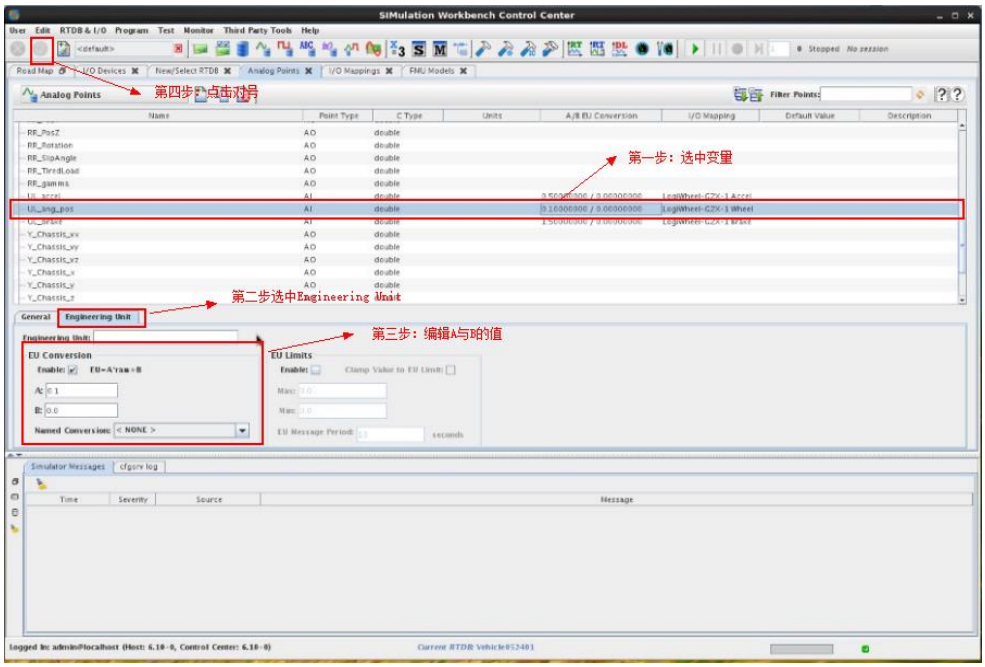


图 7-26　将方向盘的转角输入数值缩小十倍

- 接下来就是选择、配置 Tests。

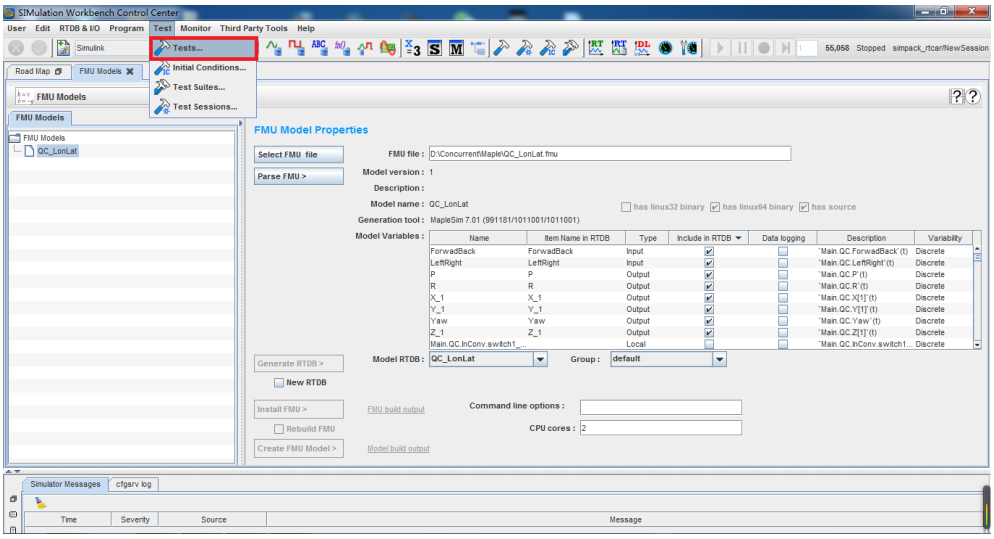


图 7-27　配置 Tests

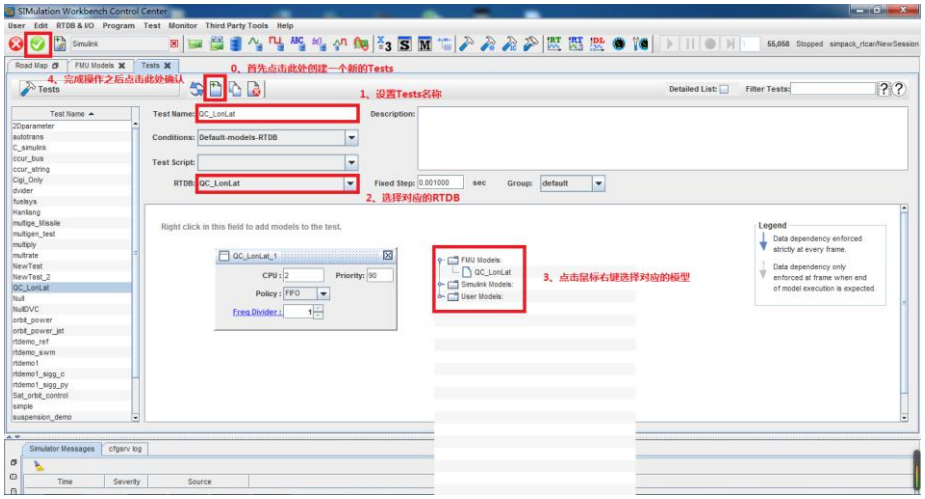
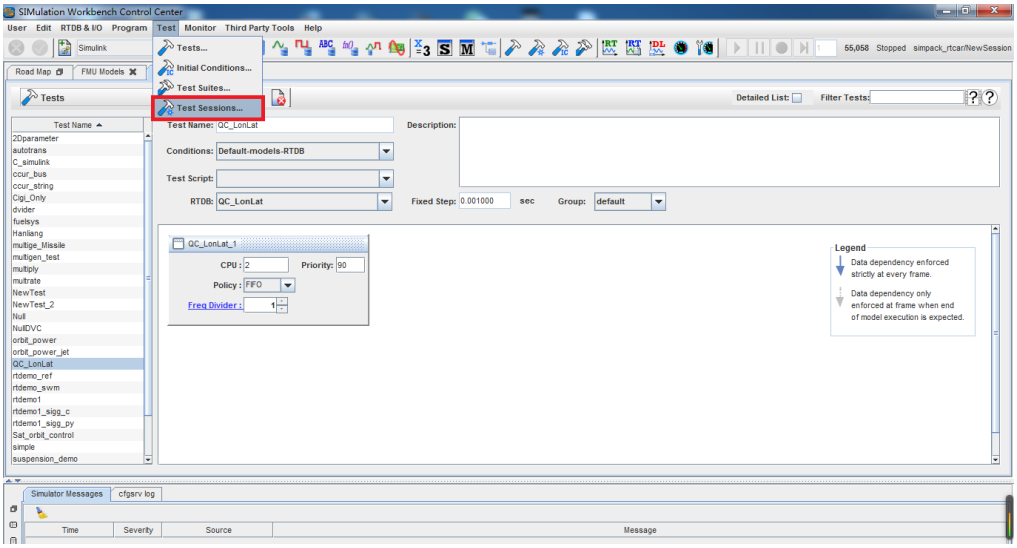


图 7-27　Test 设置

- 配置对应 Test 的 session。



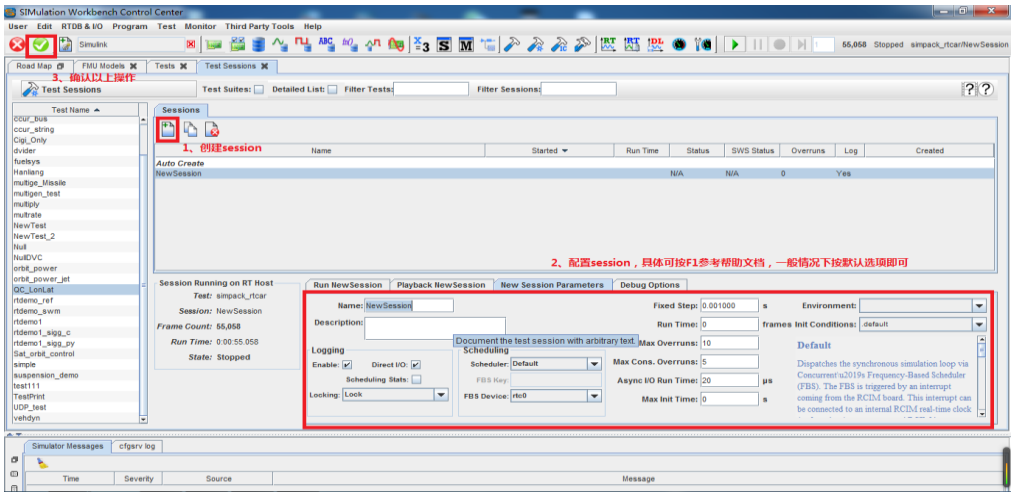


图 7-28　Test Session 设置

- 在 Test session 中运行测试实验，并可以到 RT-Viewer 中监控运行参数，在线调参等操作。

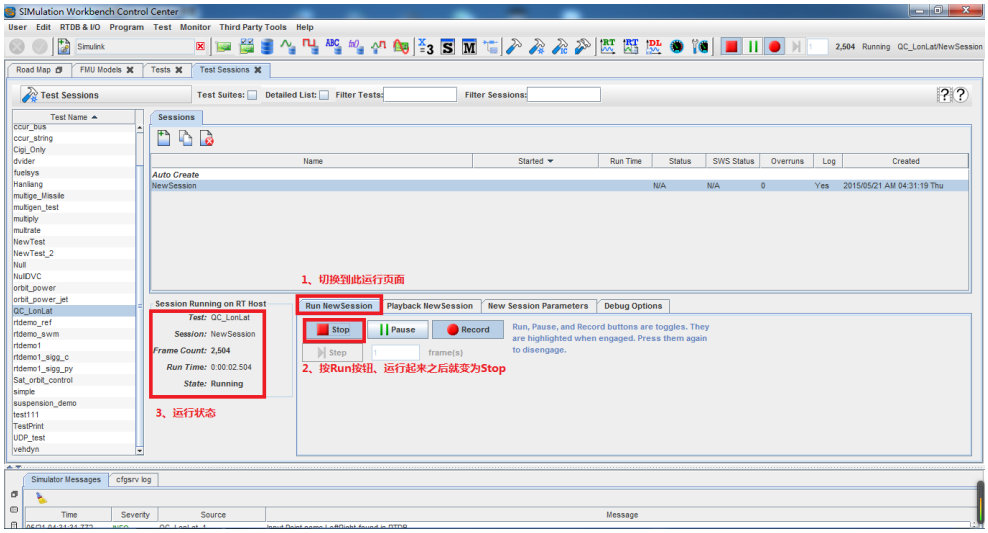


图 7-29　Test Session 中运行模型

- 点击 RT-Viewer 查看运行过程中各输入输出变量的值。

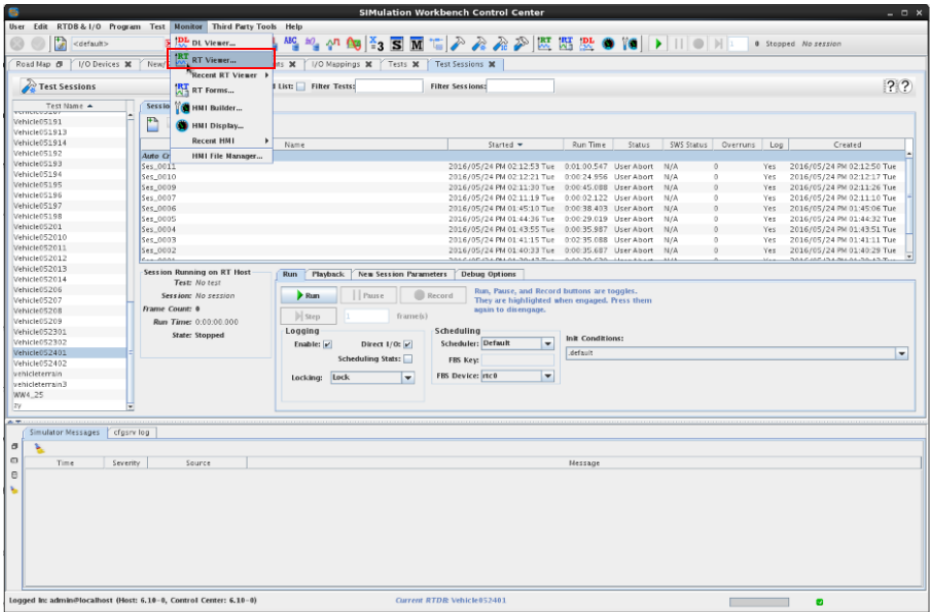


图 7-30　查看运行过程中各输入输出变量的值

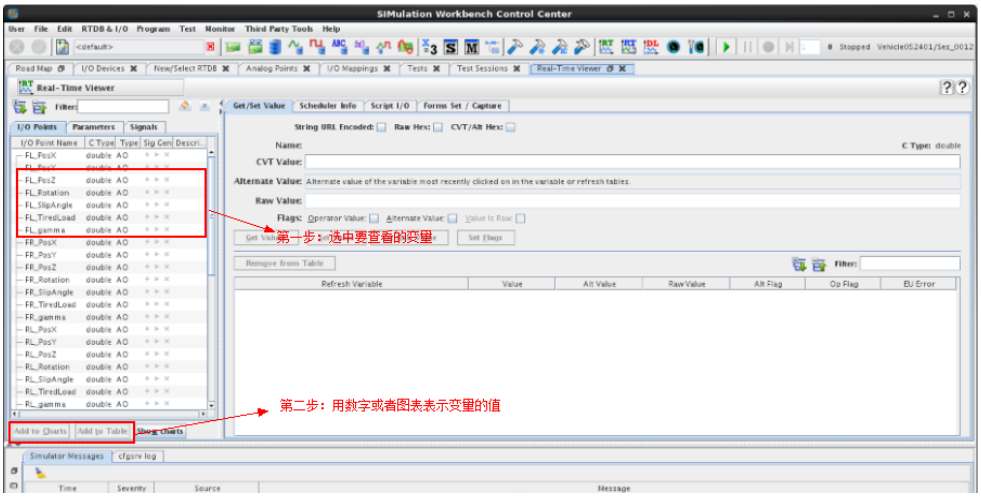


图 7-31　在模型运行过程中查看变量数值

7.7 模型运行结果对比

模型结果对比流程如图所示：

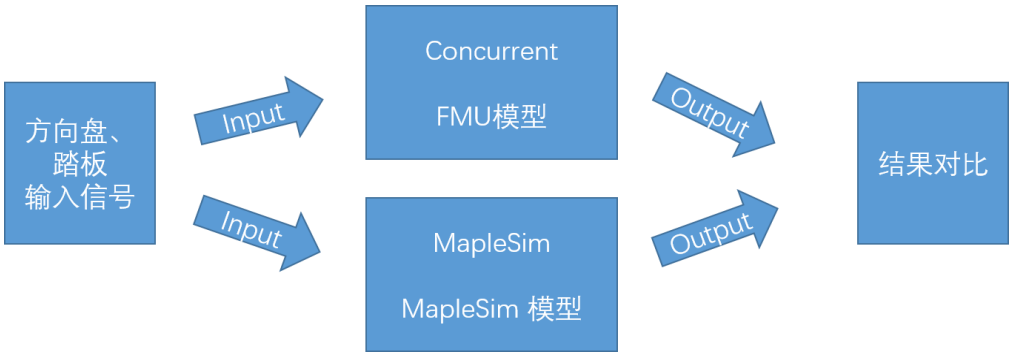


图 7-32　模型结果对比流程

采集方向盘、刹车踏板、油门踏板的信号，并且记录此时 Concurrent 输出变量的值。并把采集信号，利用 TimeTable 元件变成 MapleSim 模型输入，同时得到 MapleSim 输出变量的值。对比 Concurrent 与 MapleSim 仿真结果，判断模型运行的正确性。

第八章 电机分析

8.1 电机分析培训

1. 模型下载地址：http://www.maplesoft.com.cn/book/8.1.zip
2. 打开文件“Motor_Nao_Analysis.mism”。本次培训主要分析用于驱动 NAO 的左胳膊肩膀关节伺服电机。
3. 建立模型

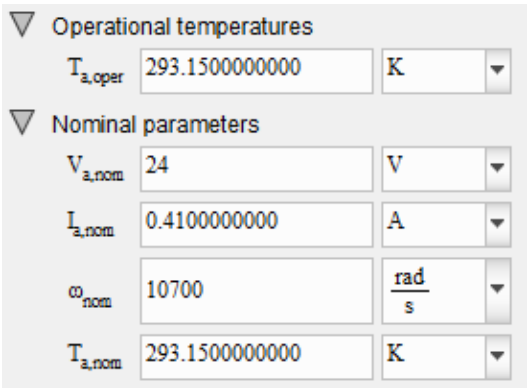
   1) Electrical > Machines > DC Machines > DC Permanent Magnet

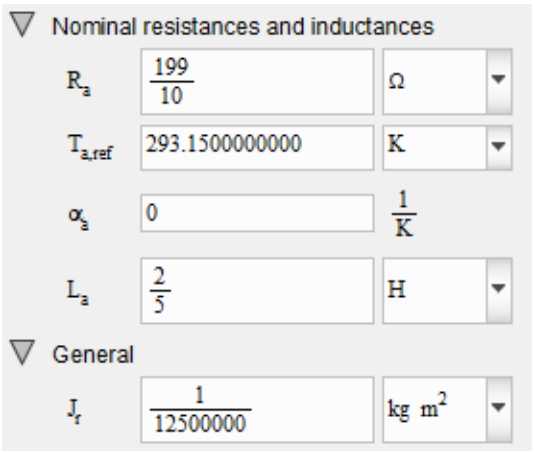
   参数为

   $T_{a,oper}$ 293.15 K；$V_{a,nom}$ 24 V；$I_{a,nom}$ 0.41 A；ω_{nom} 10700 rad/s；

   $T_{a,nom}$ 293.15 K；R_a 199/10 Ω；$T_{a,ref}$ 293.15 K；α_a 0 1/K；La 0.4 H；

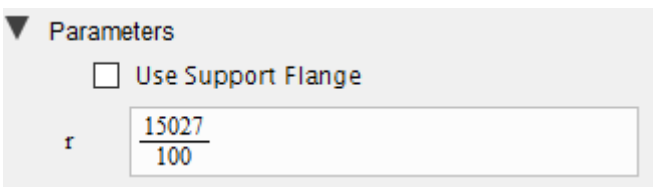
   Jr 1/12500000 kg m^2；





   2) 1-D Mechaincal > Rotational > Common > Ideal Gear

   参数：r = 15027/100



3) 1-D Mechaincal > Rotational > Sensors > Angle Sensor

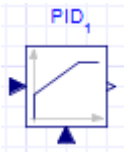
4) Electrical > Analog > Common > Ground

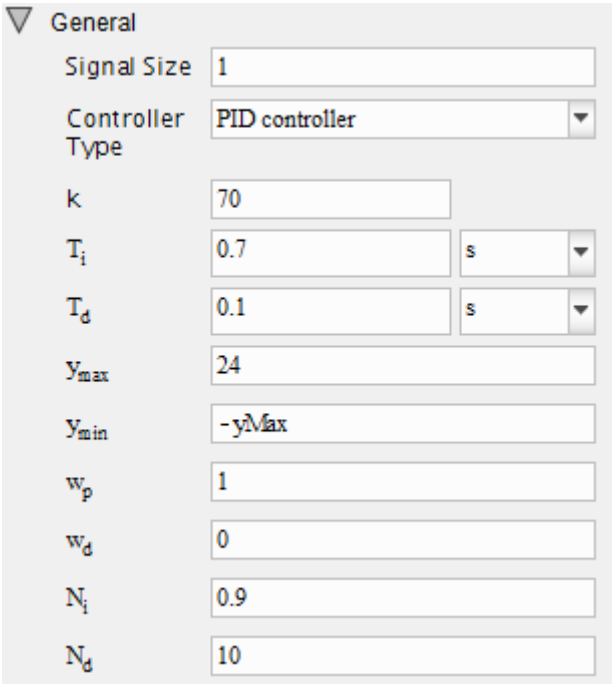
5) Electrical >Analog > Sources > Voltage > Signal Voltage

6) Signal Blocks > Controllers > Lim PID

参数：k 70 ； Ti 0.7 s； Td 0.1 s； ymax 24 ； wp 1 ； wd 0 ；

Ni 0.9 ； Nd 10 ;





7) Signal Blocks > Routing > To From Blocks > From Variable

参数：var ShoulderLeftPitchRef(Ref)

如图所示连接模型，并添加探针。

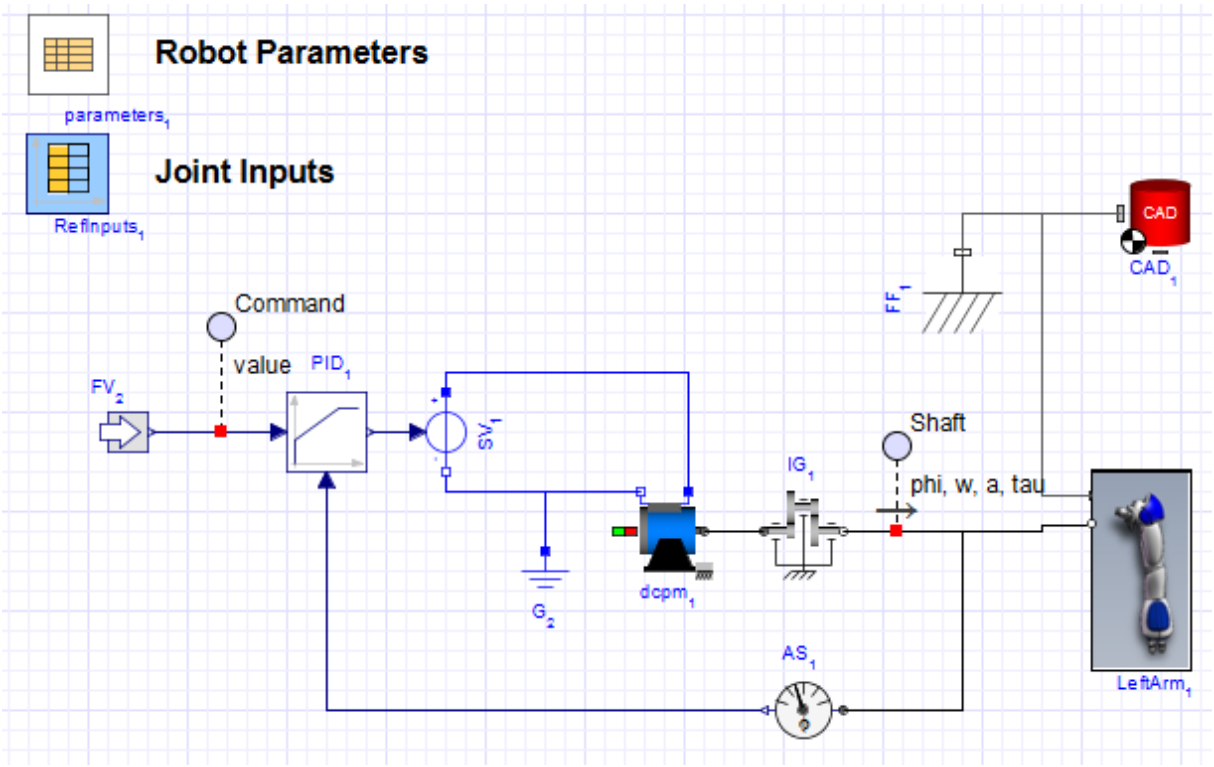
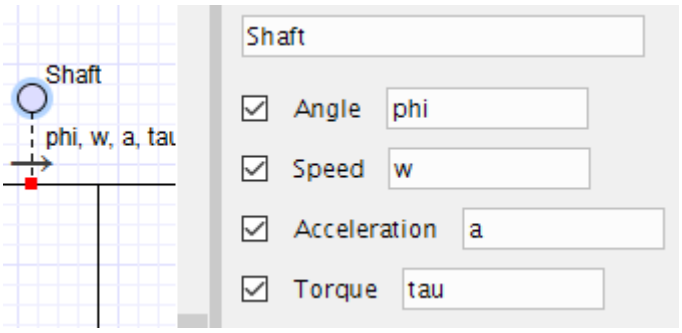


图 8-1　电机建模

共计有 2 个探针，探针 Command 和探针 Shaft，点击探针 Shaft，在右侧的参数面板中选择 Angle，Speed，Acceleration，Torque。



4. 如图所示，建立新的 Worksheet。

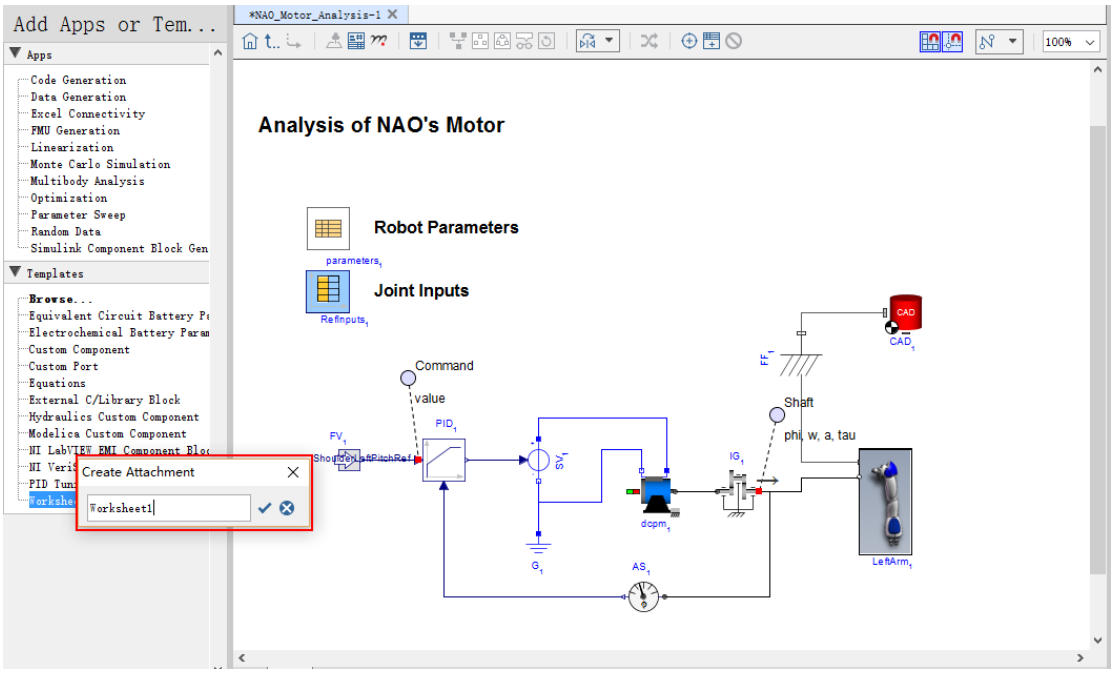


图 8-2　建立 Worksheet

1) 连接模型，加载 Maple 函数包

> A := MapleSim:-LinkModel():

> with(DocumentTools):

with(SignalProcessing):

with(plots):

2) 查看模型参数

> A:-GetParameters(allparams = true);

3) 提取模型变量，编译 MapleSim 模型

>cproc:=A:-

GetCompiledProc(params=[dcpm1_Ra=199/10,dcpm1_La=2/5,IG1_ratio=15027/100,dcpm1_Jr=1/12500000]);

4) 查看 MapleSim 模型输出

> Probes:=cproc:-GetOutputs();

5) 定义变量的值，查看输出结果

> Res:=cproc(params=[15,0.5,15,0.0000001]);

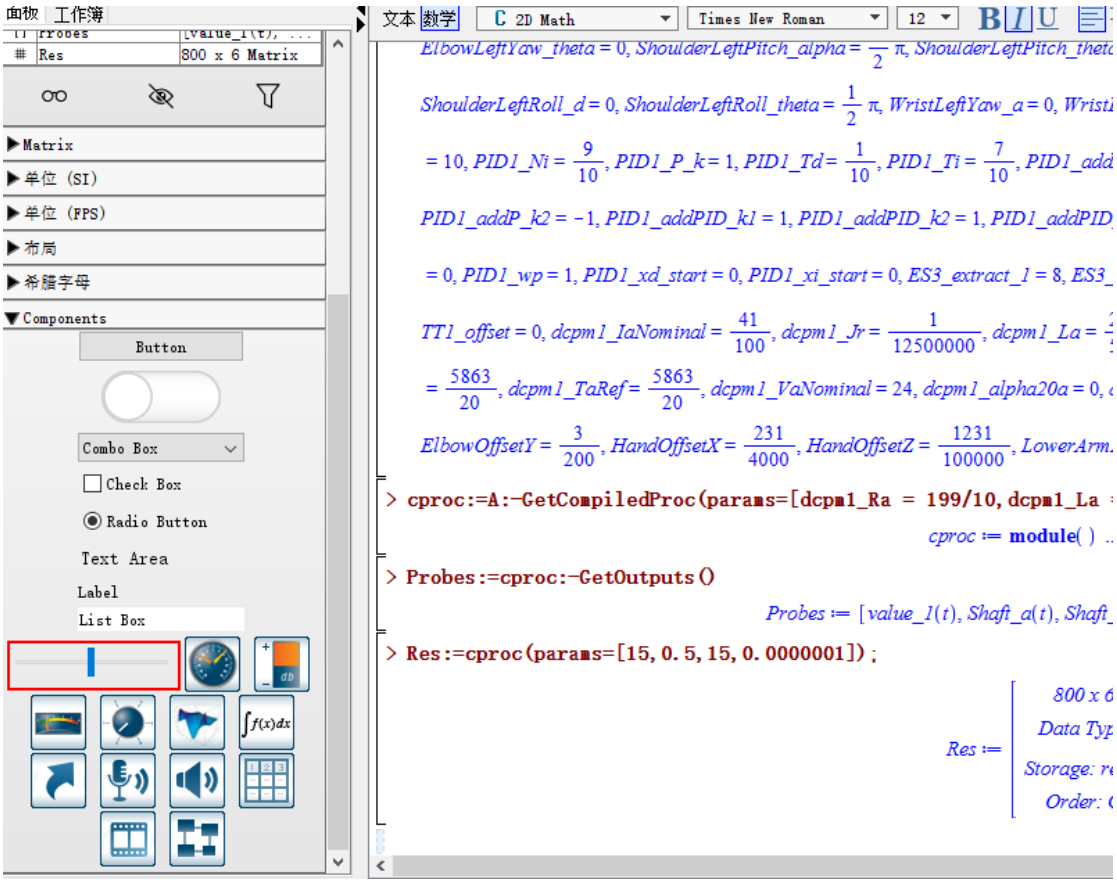


图 8-3　在 Worksheet 中执行程序

6) 定义初始值

>ParamValue1:=15;ParamValue2:=0.5;ParamValue3:=15;ParamValue4:=0.0000001;

7) 插入 4 个滑动条，滑动条元件的属性的设置，如下图所示。

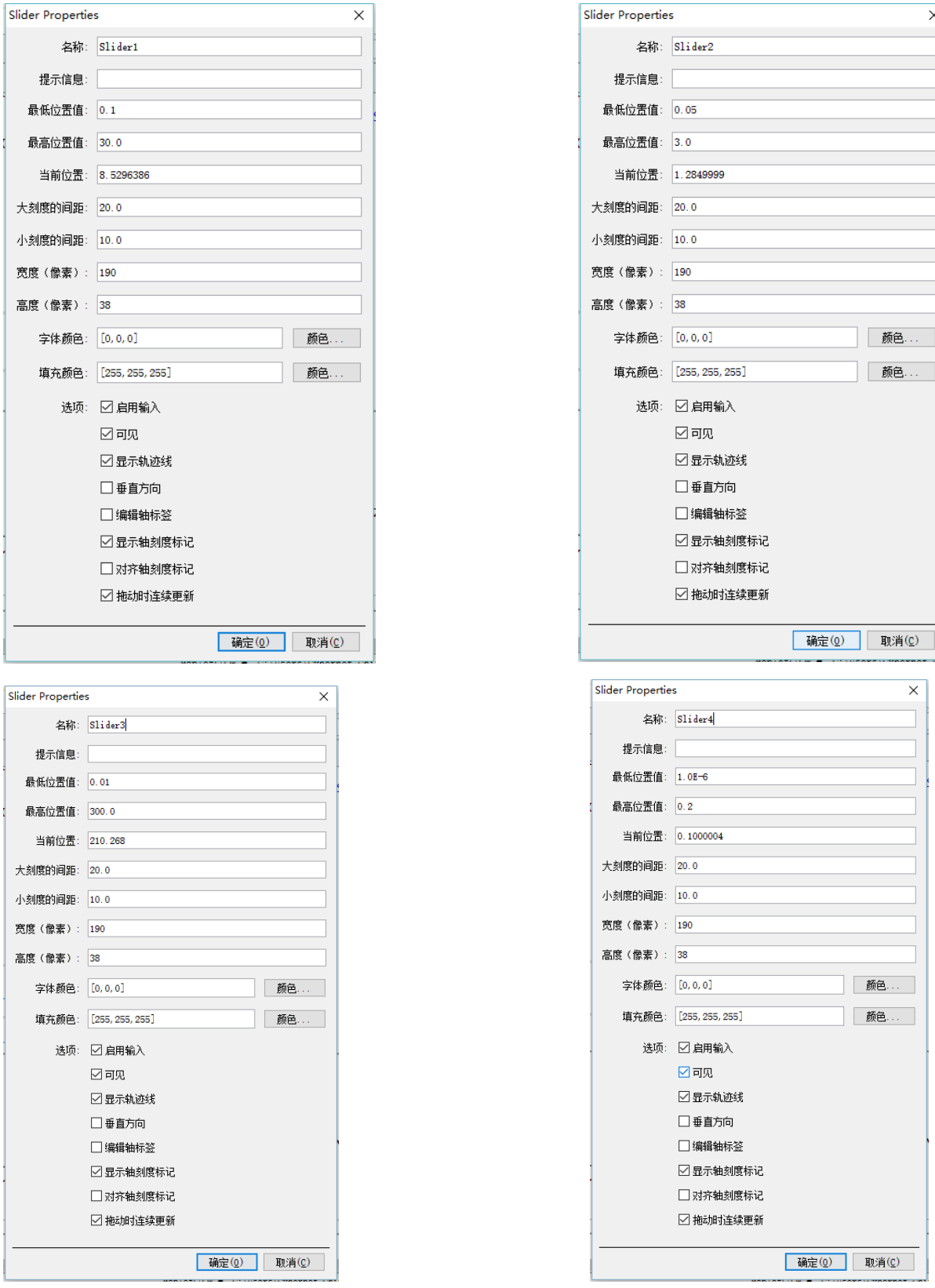


图 8-4　各个滑动条元件的属性设置

分别插入 4 个画图元件，4 个 Label。修改名称。放置位置如下图所示。

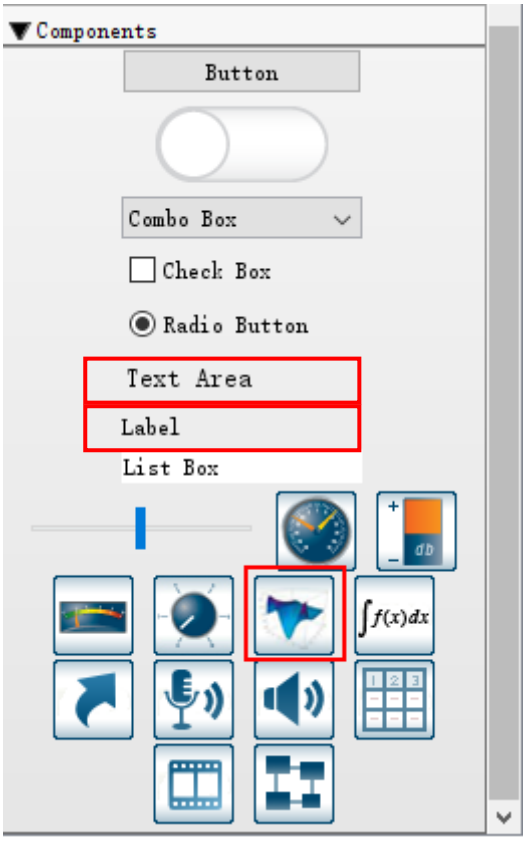


图 8-5　画图元件和 Label 元件

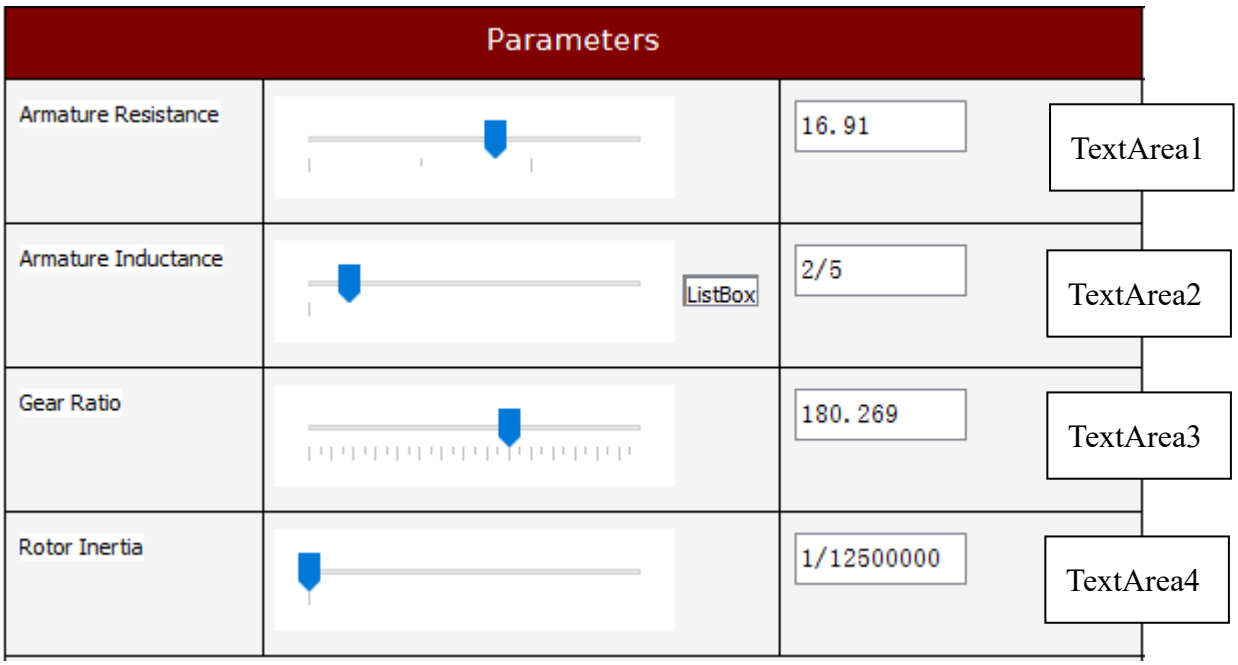


图 8-6　各个元件位置

8) 右击滑动条，编辑值改变时动作。

在组件 Slider1 输入

```
use DocumentTools in
ParamIndex := 1:
ParamValue||ParamIndex := GetProperty(Slider||ParamIndex,value);
SetProperty(TextArea||ParamIndex,value,ParamValue||ParamIndex);
Res:=cproc(params=[ParamValue1,ParamValue2,ParamValue3,ParamValue4]);
P0:=plot(Res[..,1],Res[..,2],title="Command Value"):
SetProperty(Plot0,value,display(P0));
P1:=plot(Res[..,1],Res[..,3],title="Shaft_a(t)"):
SetProperty(Plot1,value,display(P1));
P2:=plot(Res[..,1],Res[..,4],title="Shaft_phi(t)"):
SetProperty(Plot2,value,display(P2));
P3:=plot(Res[..,1],Res[..,5],title="Shaft_tau(t)"):
SetProperty(Plot3,value,display(P3));
P4:=plot(Res[..,1],Res[..,6],title="Shaft_w(t)"):
```

SetProperty(Plot4,value,display(P4));

end use;

#以上变量名和命令输入时注意区分大小写字母。

组件 Slider2 输入同样的代码，将第二句改完 ParamIndex:=2；

同理，组件 Slider3 改为 ParamIndex:=3,组件 Slider4 改为 ParamIndex:=4;

9) 改变滑动条的数值，可以看到画图组件输出仿真结果图也随之改变。从而查看改变电机参数值对电机性能的影响。

第九章 NAO 机器人建模

9.1 NAO 机器人建模

1. 模型下载地址：http://www.maplesoft.com.cn/book/9.1.zip
2. 打开文件“Hello_NAO_1.mism”。
3. 单击 Local Components 查看模型中的子系统。DH 子系统为不可动关节，DHR 子系统表示可动关节。

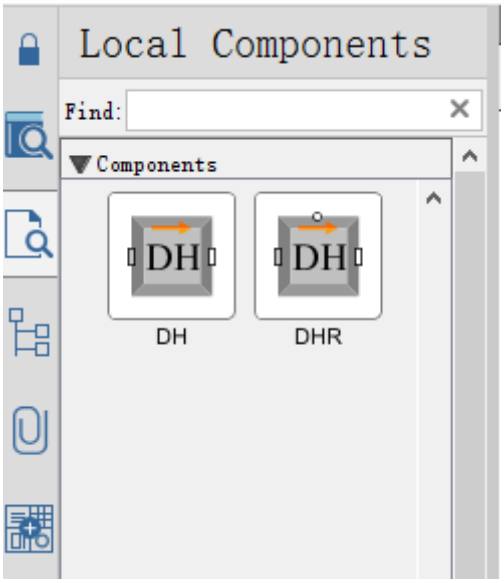


图 9-1 模型中的子系统

DH 子系统的原理如下图所示，每个连杆由 4 个参数表示，用这样的四个参数可以表示 NAO 机器人所有的连杆，对于复杂的模型用 DH 矩阵方法建模速度更快。

a_n=沿 X_n 轴，从 Z_n 移动到 Z_{n+1} 的距离;

α_n=绕 X_n 轴，从 Z_n 旋转到 Z_{n+1} 的角度;

d_n=沿 Z_n 轴，从 X_{n-1} 移动到 X_n 的距离;

θ_n=绕 Z_n 轴，从 X_{n-1} 旋转到 X_n 的角度;

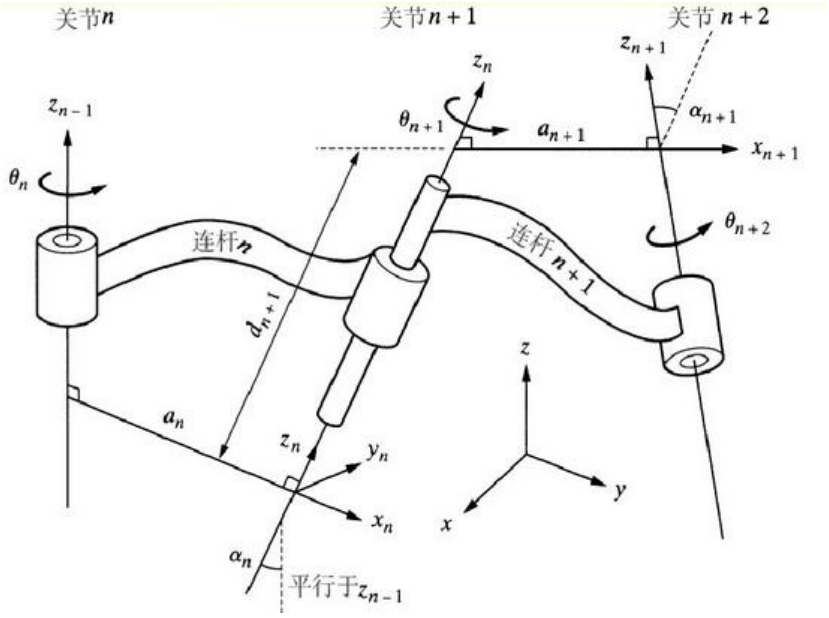


图 9-2　DH 子系统的原理图

4. 将子系统 DHR 拖入建模区域

Ctrl+G　创建子系统 子系统命名为“LeftArm”;

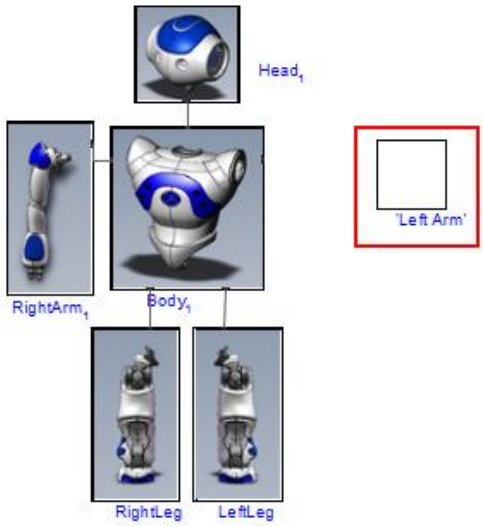


图 9-3　建立 LeftArm 子系统

双击 LeftArm 子系统，打开子系统内部

5. 再将 4 个 DHR 组件拖入 LeftArm 子系统，并 DHR 子系统如下图方式排列。并设定参数如图所示。

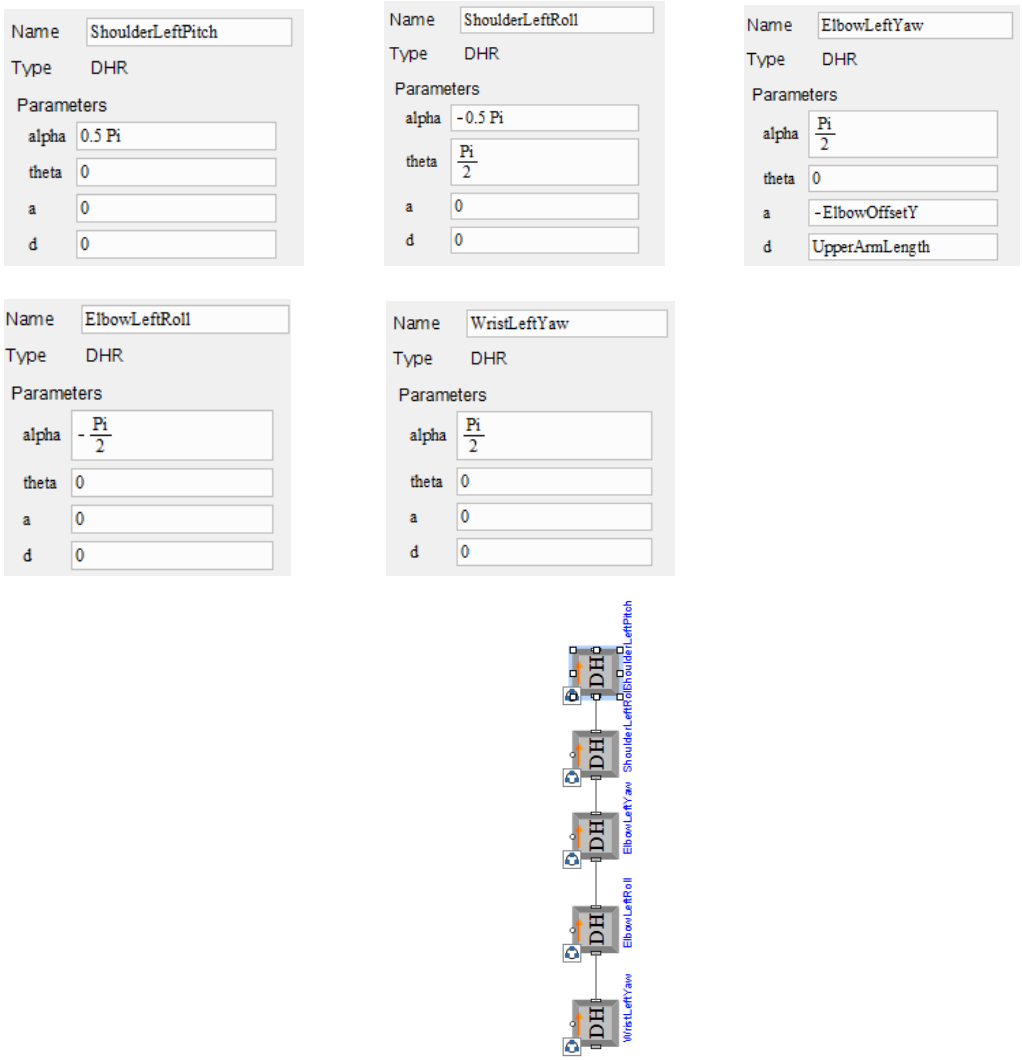


图 9-4　LeftArm 子系统中的元件参数及位置

6. Multibody > Visualization > CAD Geometry 拖入 5 个 CAD 元件到模型中并按下图所示连接模型，并定义模型参数如图所示。

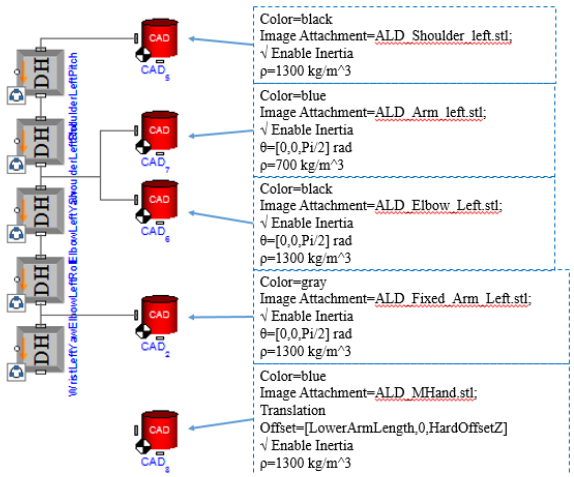


图 9-5　元件的可视化

7. Multibody > Rigid Body Frame

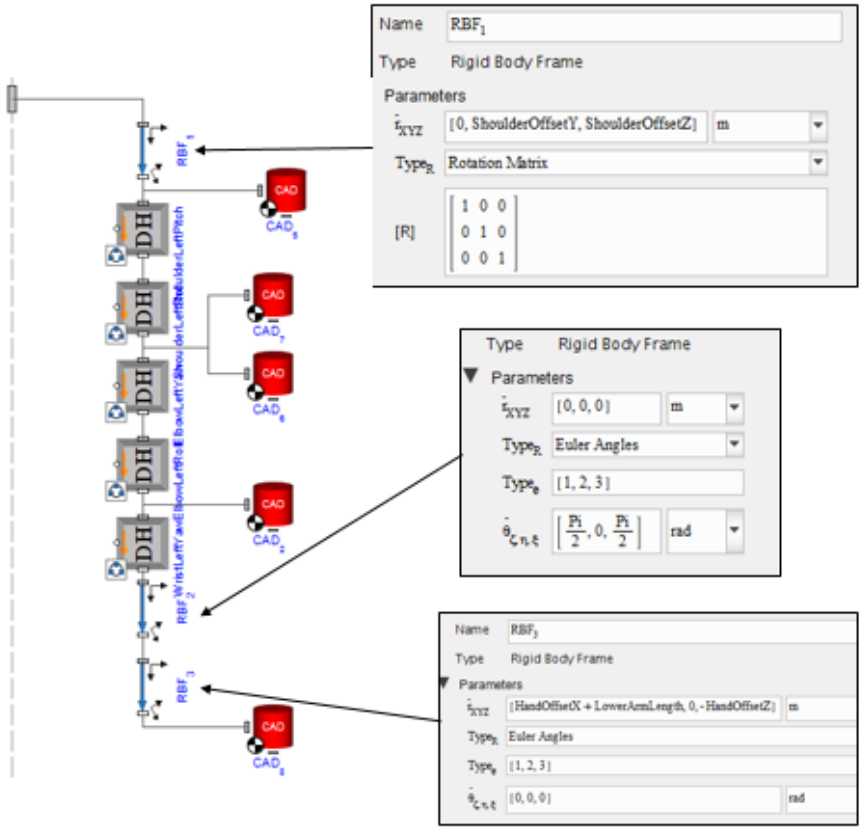


图 9-6　添加元件并设定参数

8. 1-D Mechanical > Rotational > Motion Driver > Position

拖入 5 个 Position 元件

Signal Blocks > Routing > To-From Blocks > From Variable

拖入 5 个 From Variable ，作为 Position 的驱动

Multibody > Visualization > Path Trace 拖入元件显示手臂运动轨迹并加入探针。

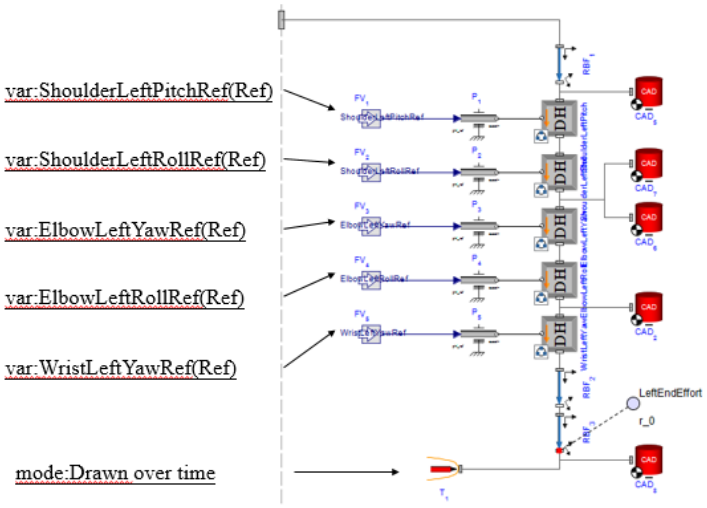


图 9-7　添加元件并修改参数

9. 点击  ，编辑子系统的图案。

10. 点击如下图标，选择 Image ——机械臂图片。

图 9-8　修改系统图标

11. 返回主菜单运行模型，查看运行结果。

第十章 跳舞机器人

10.1 跳舞机器人

1. 模型下载地址：http://www.maplesoft.com.cn/book/10.1.zip
2. 打开文件“NAO_Dances.mism”。
3. 点开子系统 Head，将两个 Component DHR 拖入建模区域。

DHR1：Name=HeadPitch; alpha=0.5Pi; theta=-0.5Pi;a=0;d=0;

DHR2：Name=HeadYaw; alpha=0; theta=0; a=0; d=0。

图 10-1 DHR 子系统

4. 2 个 Rigid Body Frame 元件 .Multibody > Bodies and Frames > Rigid Body Frame

RBF1: rXYZ=[0,0,NeckOffsetZ] m;

RBF2: θ=[0,-0.5Pi,0.5Pi] rad;

5. 2 个 CAD Geometry 元件. Multibody > Visualization > CAD Geometry

CAD_1：Image Attachment=ALD_Neck .stl ; √ Enable Inertia ; ρ=1300 kg/m^3

CAD_2：Image Attachment=ALD_Head.stl ; √ Enable Inertia ; ρ=5000 kg/m^3

6. 2 个 Position 元件. 1-D Mechanical > Rotational > Motion Driver > Position.
7. 2 个 From Variable 元件. Signal Blocks > Routing > To-From Blocks > From Variable.

FV_1：var=HeadPitchRef(Ref); n=1;

FV_2：var=HeadPitchRef(Ref); n=1;

连接元件如下图所示：

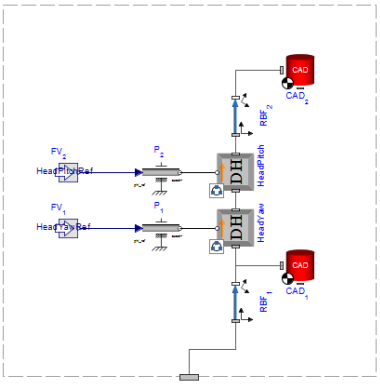


图 10-2　各个元件之间的连接

8. 点击 Return to main 回到主页面，将 $Head_1$ 与 $Body_1$ 相连。

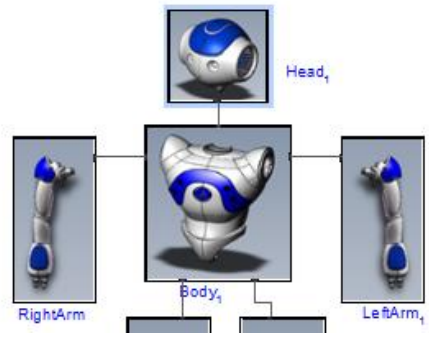


图 10-3　连接后的模型

9. 运行模型得到仿真结果。

10. 打开 Joint Inputs，查看 TT_1 元件参数。变换 data 的文件名，可以改变机器人的跳舞动作。

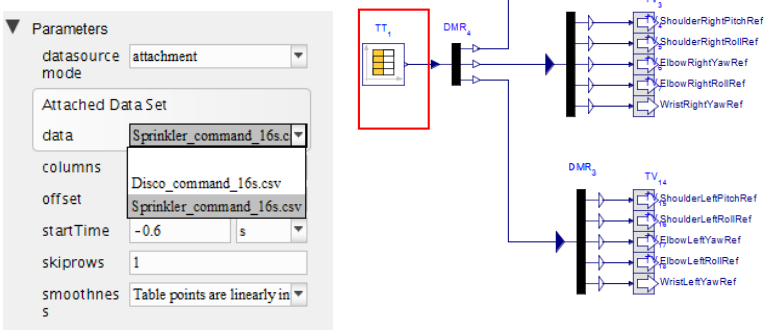


图 10-4　设定机器人跳舞动作

第十一章 自平衡车建模教程

概要

自平衡车的工作原理如下图所示，我们将使用多体机械库，电气和信号元件库中的元件，对系统进行建模与仿真：

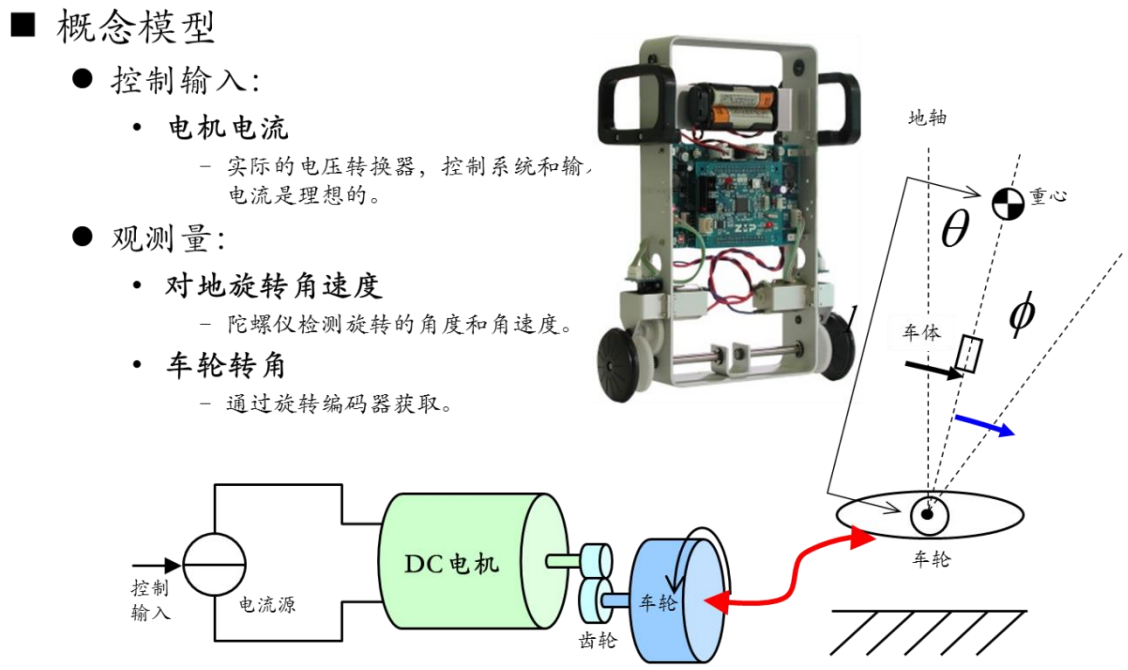


图 11-1　平面曲柄滑块机构原理图

在练习中，我们将完成下列任务：

1. 创建自平衡车结构模型；
2. 定义子系统参数；
3. 添加驱动和传感器；
4. 模型的线性化；
5. 控制设计工具箱进行控制设计；
6. 系统仿真与 3D 动画。

模型下载地址：http://www.maplesoft.com.cn/book/11.1.zip

11.1 创建自平衡车结构模型

自平衡车的结构如下图所示，在 MapleSim 中，使用多体元件库中的元件构建这个结构，如下图所示：

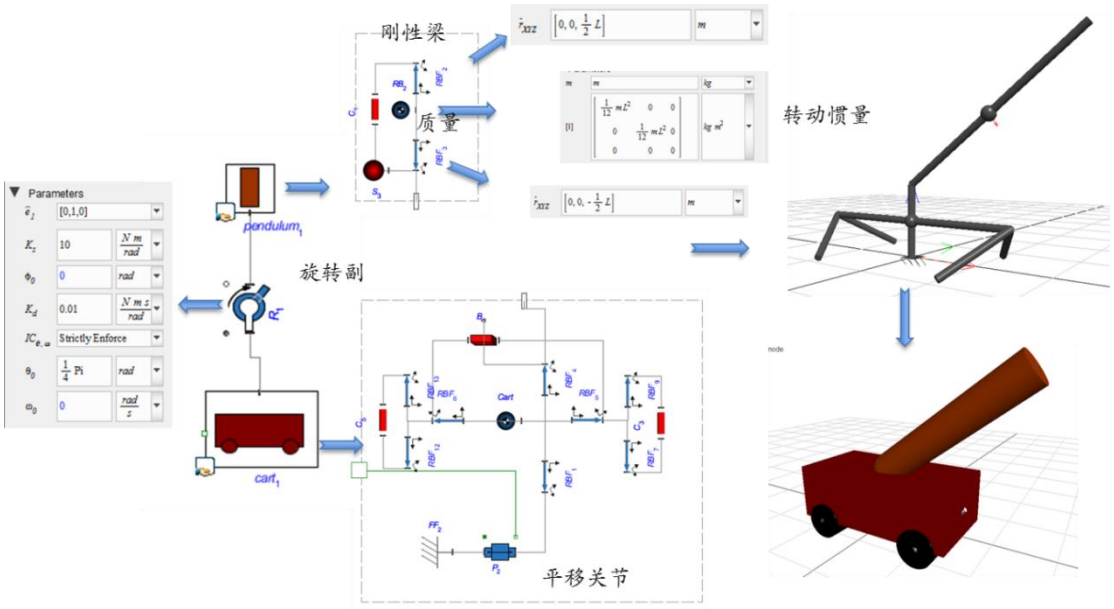


图 11-2　使用 Multibody 元件库建立模型

1. 创建车体模型

首先，设定重力方向如下图所示：

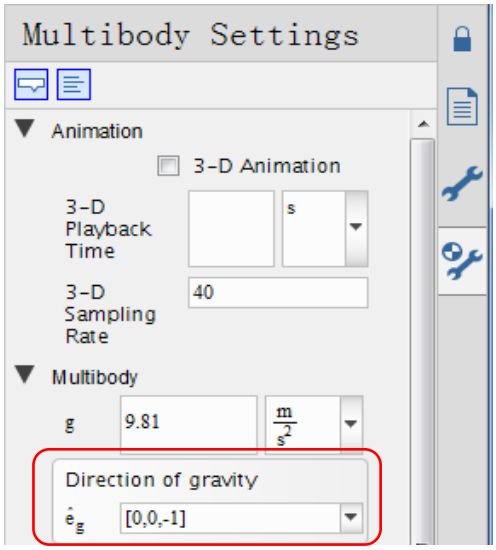


图 11-3　设置重力方向

从多体元件库中，拖入多体元件，如下图所示：

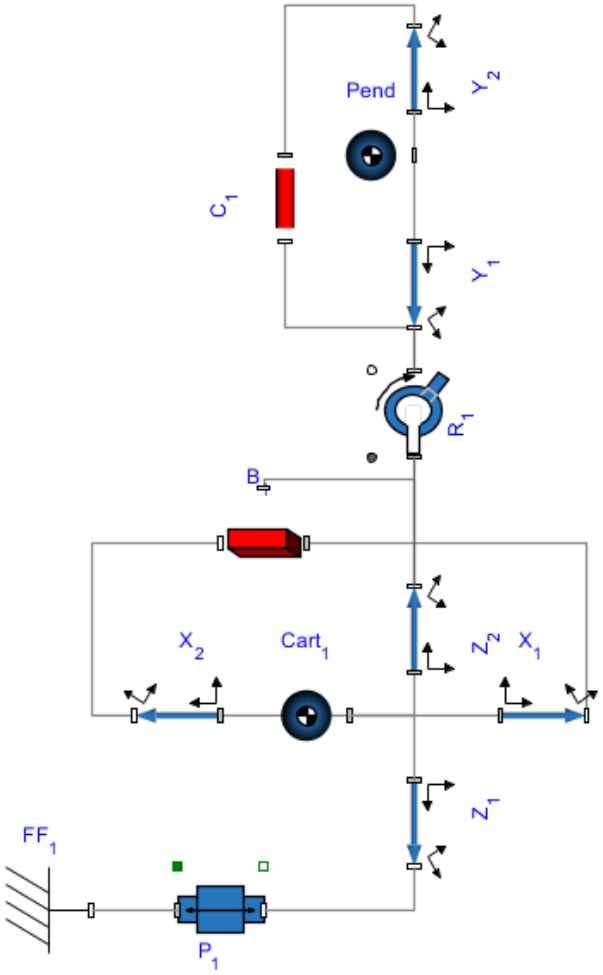
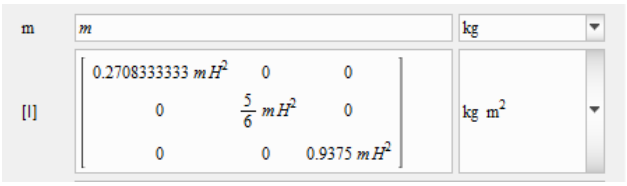
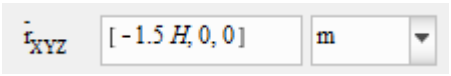
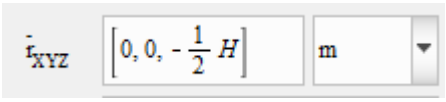
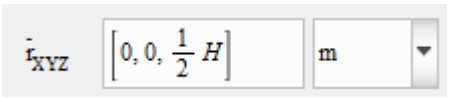
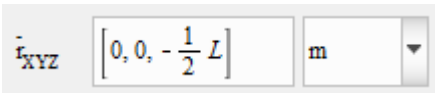
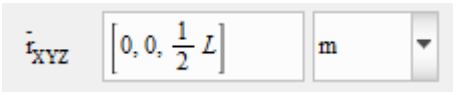
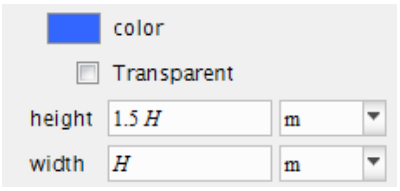
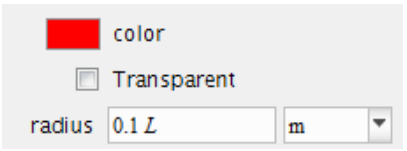
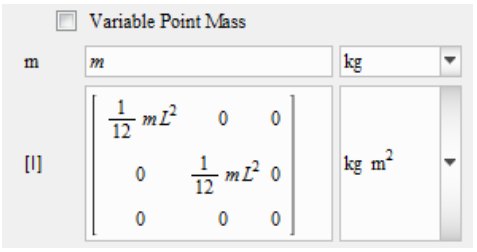


图 11-4　各个元件位置及连接

各元件的位置及设定参数如下表所示：

表 11-1 模型中的元件

名称	位置	参数设置
FF1	Multibody > Bodies and Frames > Fixed Frame	
P1	Multibody > Joints and Motions > Prismatic	$\hat{e}_1$ [1,0,0]

Cart	Multibody > Bodies and Frames > Rigid Body	
X1	Multibody > Bodies and Frames > Rigid Body Frame	
X2	Multibody > Bodies and Frames > Rigid Body Frame	
Z1	Multibody > Bodies and Frames > Rigid Body Frame	
Z2	Multibody > Bodies and Frames > Rigid Body Frame	
Y1	Multibody > Bodies and Frames > Rigid Body Frame	
Y2	Multibody > Bodies and Frames > Rigid Body Frame	
B1	Multibody > Visualization > Box Geometry	
C1	Multibody > Visualization > Cylinerical Geometry	
Pend	Multibody > Bodies and Frames > Rigid Body	

R1	Multibody > Joints and Motions > Revolute	$\hat{e}_1$ [0,1,0] K_s 0 N m/rad Φ_0 0 rad K_d 0.1 N m s/rad $IC_{\theta,\omega}$ Strictly Enforce θ_0 0.25 rad ω_0 0 rad/s

选中车体部分，单击右键，创建车体子系统，命名为 Cart：

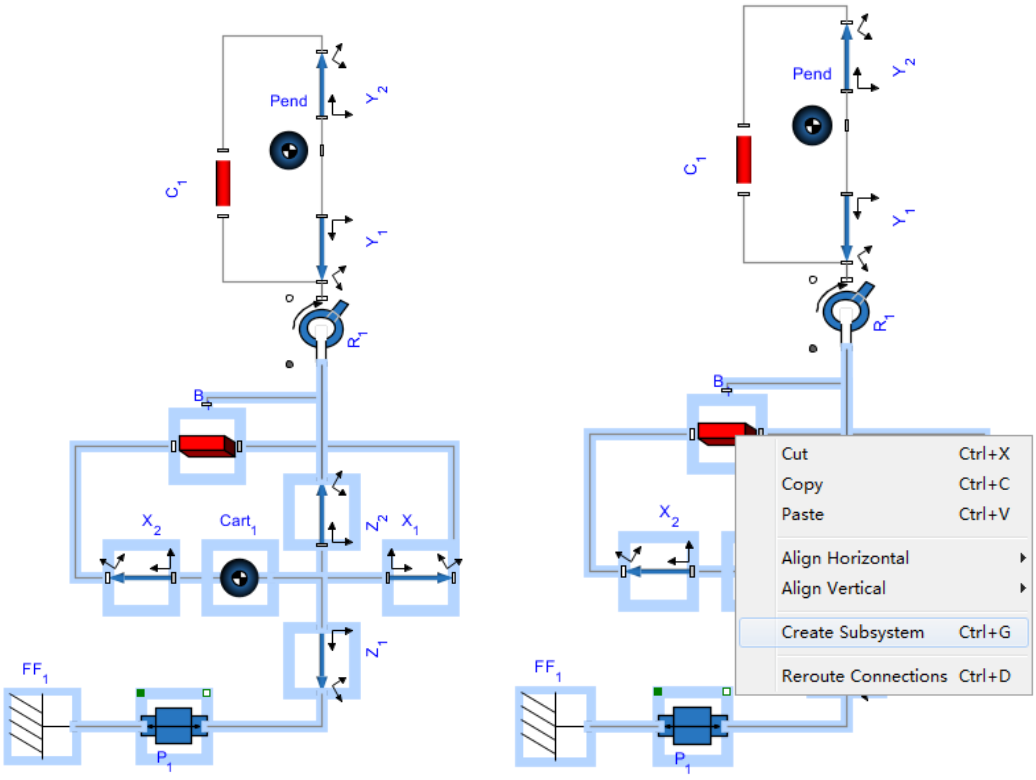


图 11-5　创建车体子系统

选择倒立摆部分，创建子系统，命名为 Pendulum，得到模型，如下图所示：

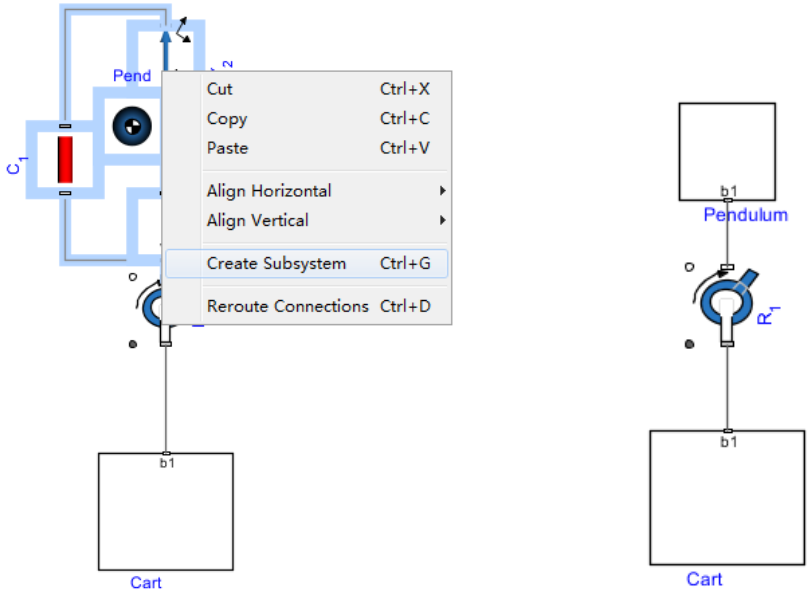


图 11-6　创建倒立摆子系统

11.2 定义子系统参数

进入 Cart 子系统，点击参数按钮（ ），新建参数 H 和 m，如下图示：

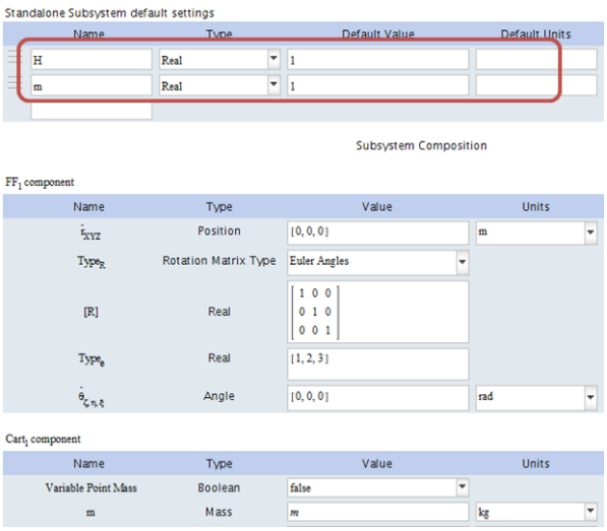


图 11-7　定义 Cart 子系统的参数

同样，对 Pendulum 子系统，点击参数按钮（ ），设定参数 L 和 m，如下图：

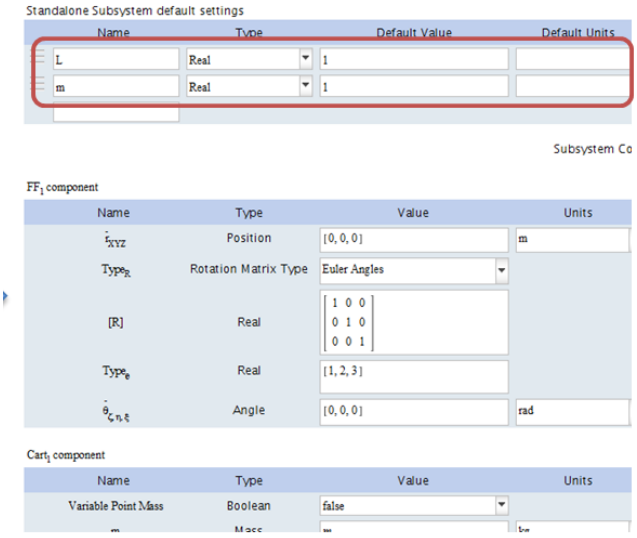


图 11-8　定义 Pendulum 子系统的参数

选中子系统 Pendulum，设定 L=3， m=2。

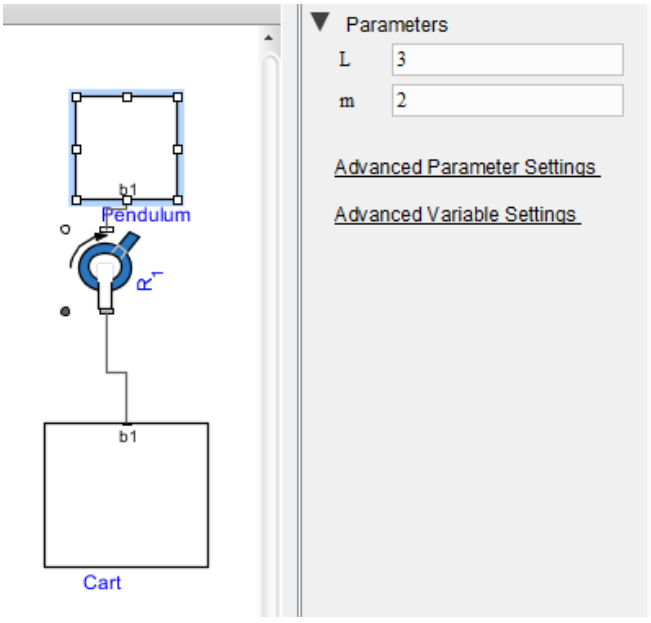


图 11-9　设定 Cart 子系统的参数值

选中子系统 Cart，设定 H=1，m=10。

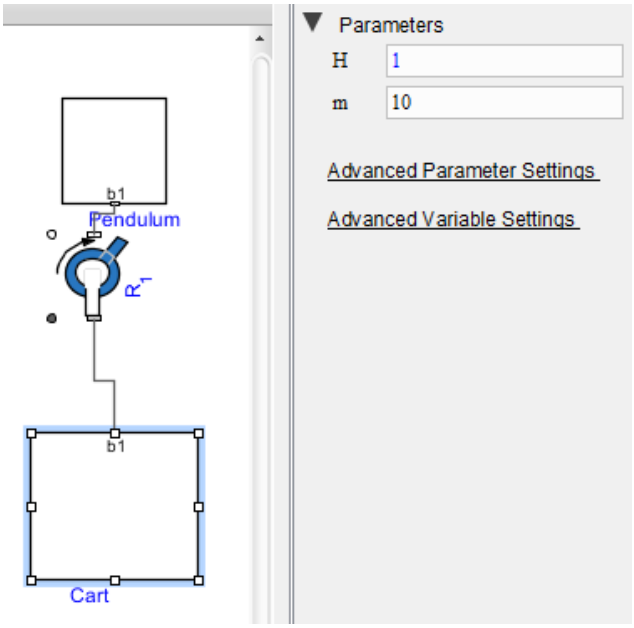


图 11-10　设定 Pendulum 子系统的参数值

11.3 添加驱动和传感器

对平移运动副的驱动点，创建一条连接线到子系统虚线框的边缘，上下移动位置，出现一个小方块时，成功创建接口，如下图所示：

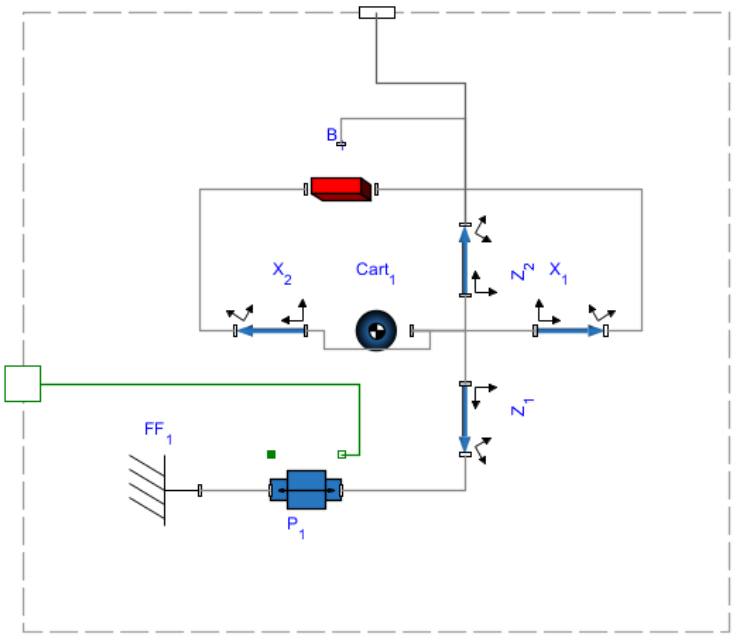


图 11-11　在车体子系统中创建接口

在上一节创建的模型上增加驱动和传感器，并创建子系统 Inverted Pendulum，如下图所示：

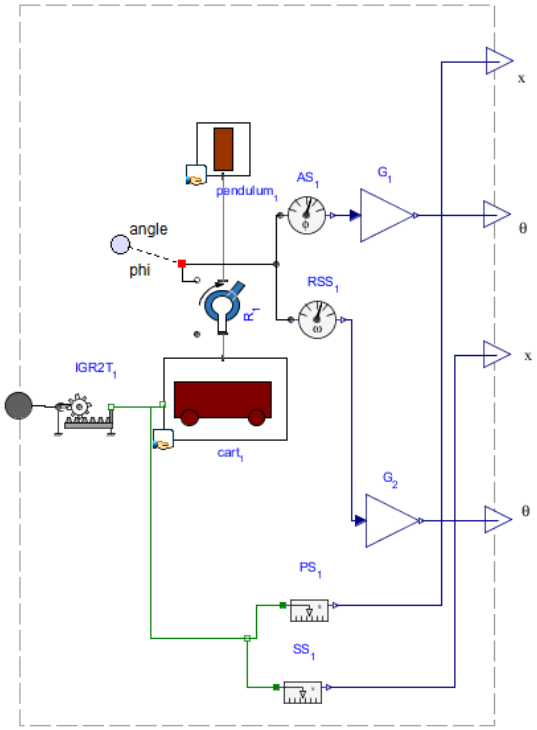


图 11-12　Inverted Pendulum 子系统

各元件的位置和参数如下表所示：

表 11-2 添加在模型中的元件

名称	位置	参数设置
IGRT1	1-D Mechanical > Translational > Common > Ideal Gear R 2 T	Use Support R Use Support T ratio 1 rad/m
G1，G2	Signal Blocks > Common > Gain	Signal Size 1 k -1.0
AS1	1-D Mechanical > Rotational > Sensors > Angle Sensor	toUnit radian

RSS1	1-D Mechanical > Rotational > Sensors > Rotational Speed Sensor	toUnit rad/s
PS1	1-D Mechanical > Translational > Sensors > Position Sensor	toUnit m
SS1	1-D Mechanical > Translational > Sensors > Speed Sensor	toUnit m/s

在子系统外部增加元件如下图所示，并创建为 Plant 的子系统，如下图：

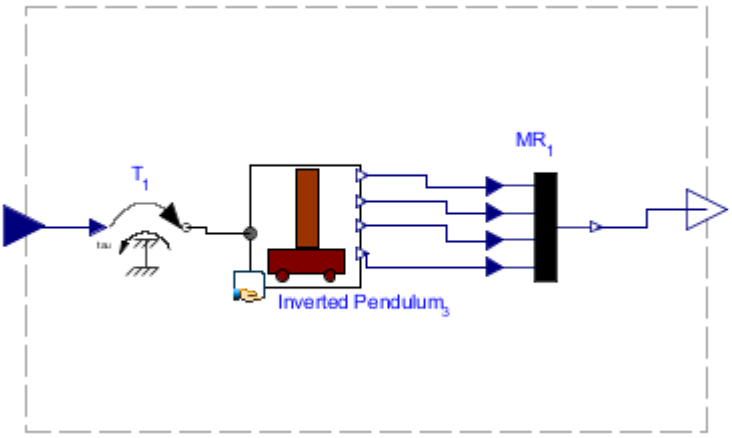


图 11-13　Plant 的子系统

11.4　模型的线性化

完成下面的模型，添加一个输入信号，并用探针测量输出结果。

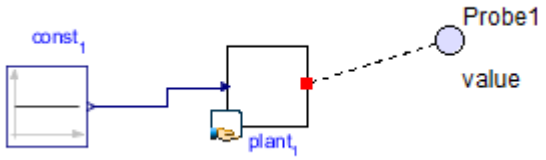


图 11-14 完整的模型

按下左侧项目面板上的  按钮，打开附属文件夹。从 Apps 下拉菜单中选择 Linerization，然后双击。

系统自动运行 Maple，并打开一个线性化分析模板。在下拉列表中选择 Plant1，会显示模型的结构。

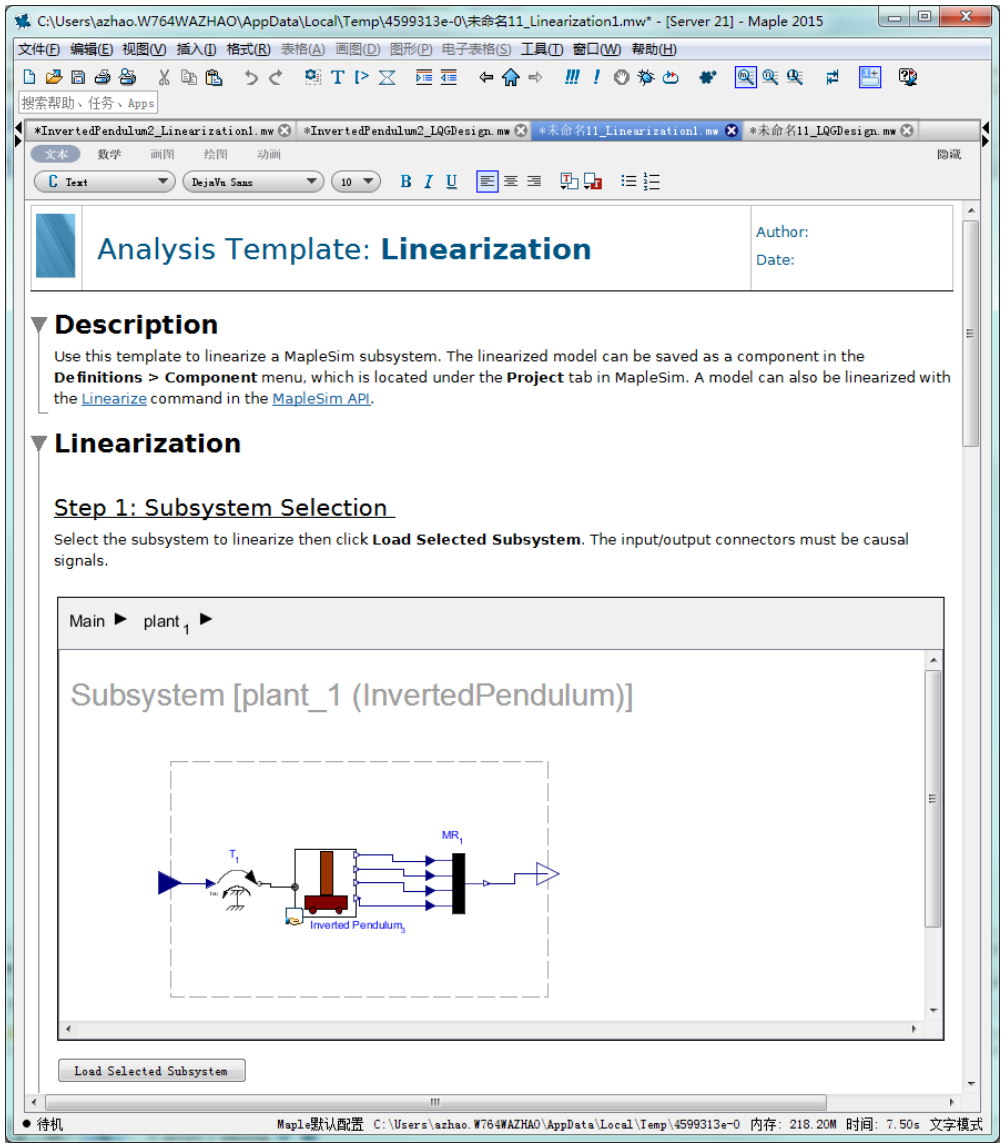


图 11-15　Linearization 窗口

单击 Load Selected Subsystem 按钮，系统会对模型进行自动分析，得到如下结果：

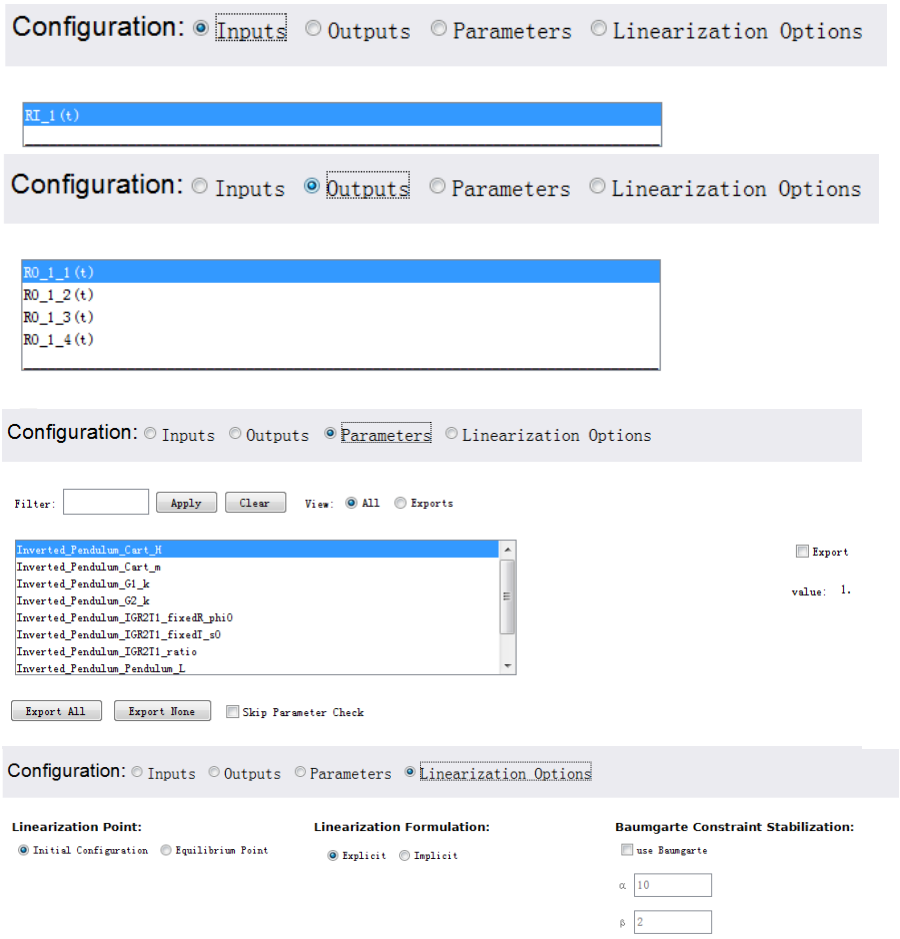


图 11-16　自平衡车模型的输出、输入、参数及线性化选项

第三步，点击 Linearize 按钮，将会得到系统的方程。

第四步，在 Component Name 中输入 linsys，并输入简单描述，单击 Create 按钮。

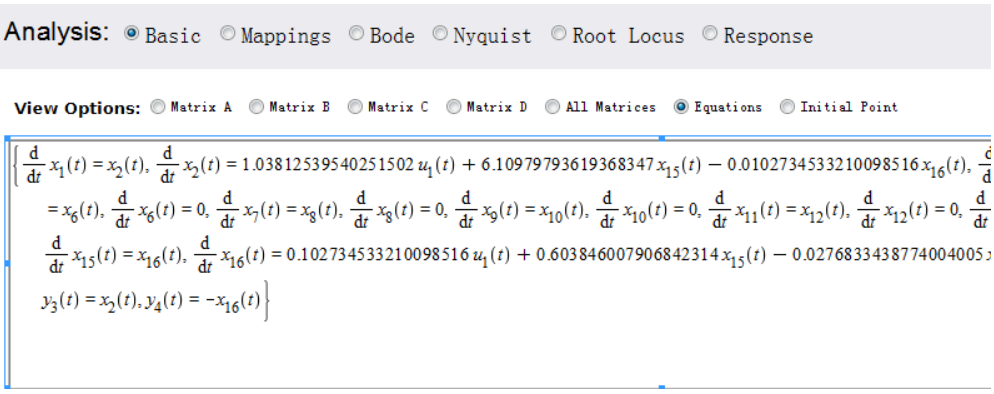


图 11-17　模型的数学方程

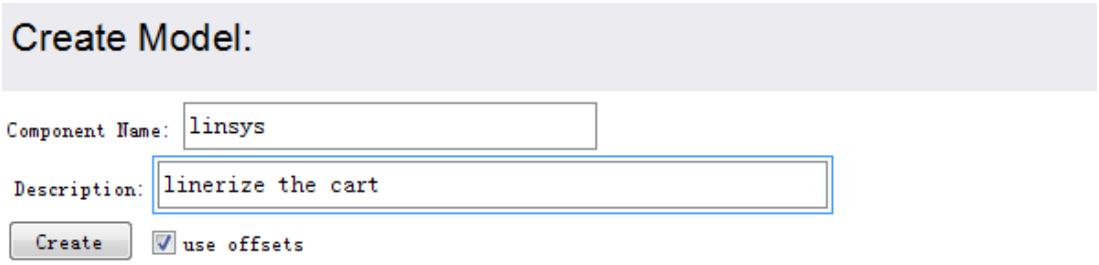


图 11-18　线性化元件的名称及描述

在 MapleSim 左侧将会出现一个新的元件 linsys，如下图所示。

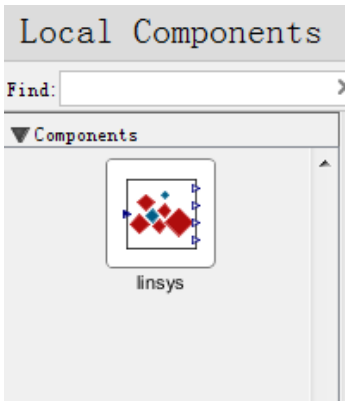


图 11-19　建立的线性化元件位置

11.5 控制设计工具箱进行控制设计

在 MapleSim 左侧项目选项卡的 Add Apps and Templates( )中的 Browse，导入 LQGDesign.mw 文件，如下图所示。

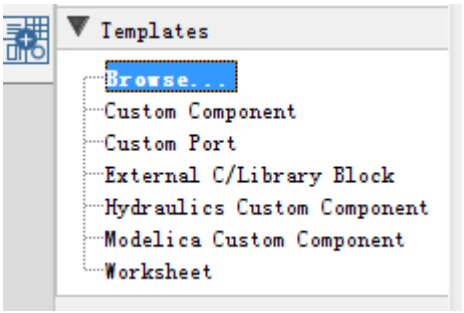


图 11-20　导入 LQGDesign.mw 文件

导入后，双击该文件名，打开控制设计工具箱模板，如下图所示。

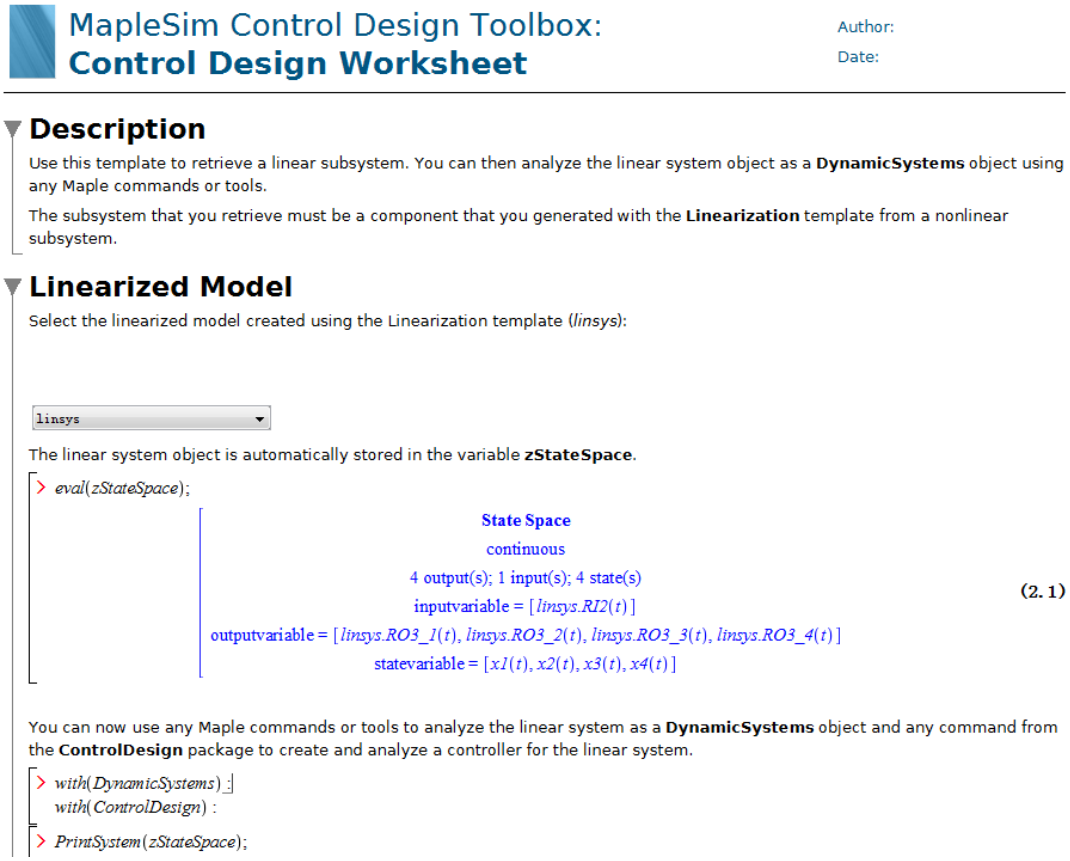


图 11-21　LQGDesign 模板

运行模板中的 Maple 命令，最终模板会自动生成一个 LQG 控制器。

LQR Design

The Q and R matrices for the LQR controller can be computed using *ComputeQR* command if the desired closed-loop time constant is sufficient performance criteria, for example:

```
> ComputeQR(zStateSpace, 1);
```

$$Record\left(Q = \begin{bmatrix} 1. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 1. \end{bmatrix}, R = \begin{bmatrix} 0.0324391509703009 \end{bmatrix}\right) \quad (3.1)$$

The Q and R matrices for our LQR controller are defined as follows:

```
> Qmat := Matrix(4, 4, {(1, 1) = 1, (1, 2) = 0, (1, 3) = 0, (1, 4) = 0, (2, 1) = 0, (2, 2) = 0, (2, 3) = 0, (2, 4) = 0, (3, 1) = 0, (3, 2) = 0, (3, 3)
      = 1, (3, 4) = 0, (4, 1) = 0, (4, 2) = 0, (4, 3) = 0, (4, 4) = 0});
  Rmat := Matrix(1, 1, {(1, 1) = 1})
```

$$Qmat := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Rmat := \begin{bmatrix} 1 \end{bmatrix} \quad (3.2)$$

The gain value, K, for the LQR controller is computed as follows:

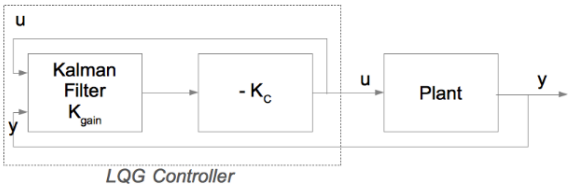
```
> K := LQR(zStateSpace, Qmat, Rmat);
```

$$K := \begin{bmatrix} -1. & 280.878509395939 & -5.85352136845337 & 119.514540002435 \end{bmatrix} \quad (3.3)$$

图 11-22　运行模板中的命令

LQG Design

A Linear Quadratic Gaussian (LQG) controller can be developed by combining the LQR controller with a Kalman observer as shown:



LQG Controller

It is also possible to include a reference gain in the LQG controller above, if desired.

To begin the LQG design, we first compute the *LQR* gain:

```
> Kc := LQR(zStateSpace, Qmat, Rmat);
```

$$Kc := \begin{bmatrix} -1. & 280.878509395939 & -5.85352136845337 & 119.514540002435 \end{bmatrix} \quad (4.1)$$

The additional G and H matrices are used to define the stochastic state-space form

$x' = A \cdot x + B \cdot u + G \cdot w$

$y = C \cdot x + H \cdot w + v$

where w and v represent white noise.

```
> G := LinearAlgebra:-IdentityMatrix(4);
  H := LinearAlgebra:-ZeroMatrix(4);
```

$$G := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.2)$$

图 11-23　生成 LQG 控制器

在 Local Component 选项卡下，将生成一个 LQGController 控制器。

将该控制器拖放到模型窗口中，并连接

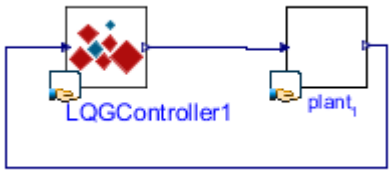


图 11-24　控制器与子系统连接

11.6 系统仿真

运行仿真按钮，可以得到仿真结果，并从 3D 动画窗口中看到仿真的动画。

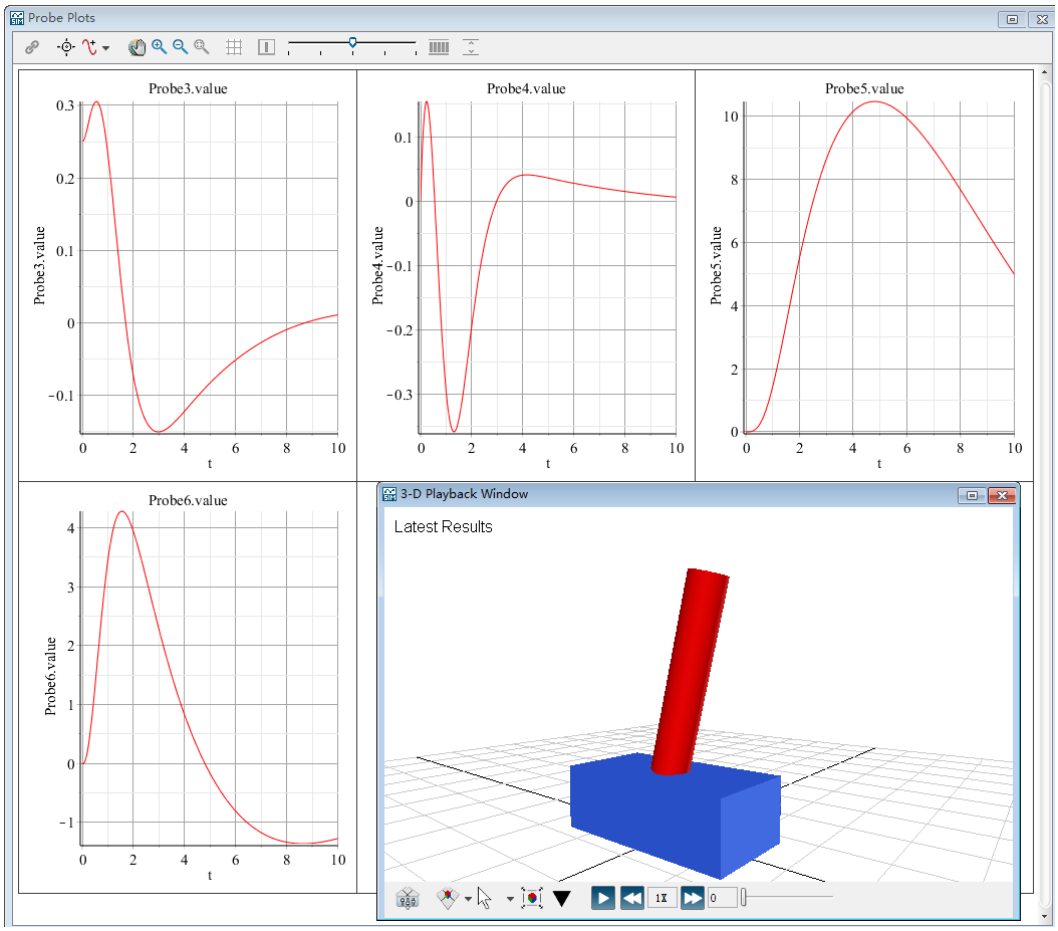


图 11-25　仿真结果

为验证控制效果，可以施加一个外力作为干扰，来验证控制器。

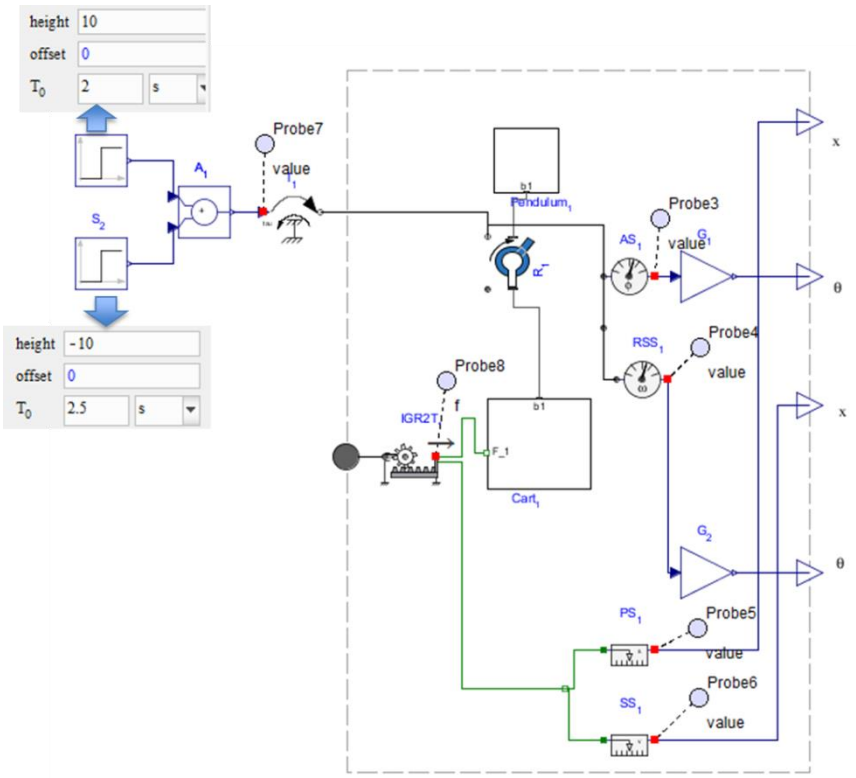


图 11-26　添加干扰元件

修改后可以看下面的仿真结果，同时从动画中可直观的看到效果。

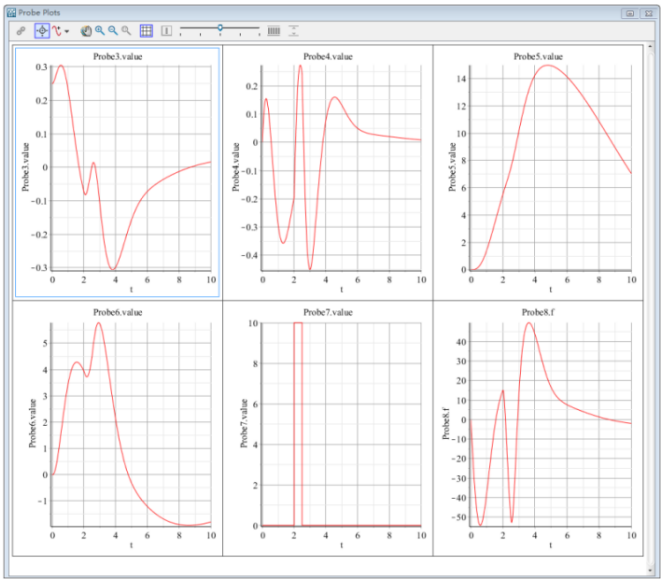


图 11-27　添加干扰后的仿真结果

第十二章 STEWART 平台运动控制

概要

1. Stewart 机构原理介绍：由六个相同的分支和上下平台组成的并联机构，分支通常由 S-P-S（球副 S-移动副 P-球副 S）或 U-P-S（万向铰 U-移动副 P-球副 S）组成。

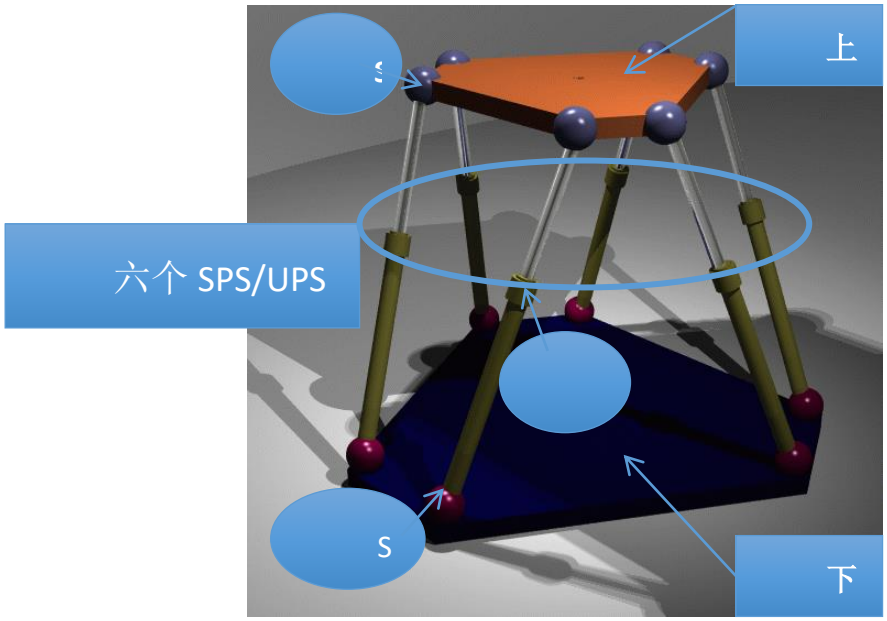


图 12-1 Stewart 三维结构图

上下平台坐标位置如下图，下平台 B_i (i=1…6) 上平台 bi (i=1…6), 其中 $B_1,B_3,B_5(b_1,b_3,b_5)$分别间隔 120 度，$B_2,B_4,B_6(b_2,b_4,b_6)$分别间隔 120 度。由于上平台的 b_1,b_6 坐标值和下平台 B_3，B_4 坐标值容易写出，其余坐标点以此为基础通过旋转向量的方式获得。

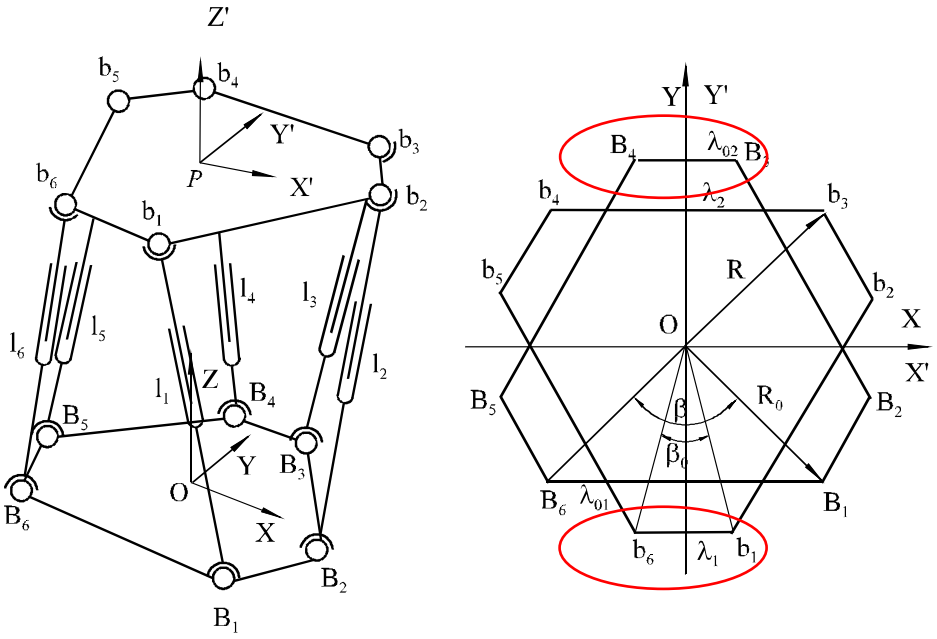


图 12-2　Stewart 结构示意图

2. 参数化的建模方式：一个模型可适用于所有的 Stewart 机构，便于修改和优化等。

如下图 Stewart 模型 1：上下平台的半径相等，夹角为 0，Stewart 模型 2：下平台与上平台半径不相等，夹角都不为 0。

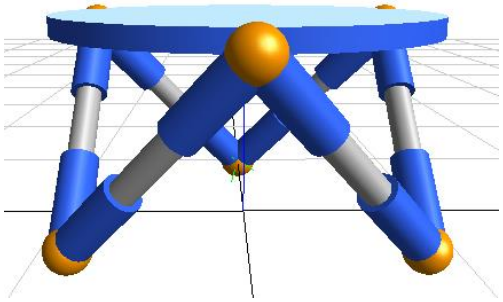


图 12-3　Stewart 模型 1

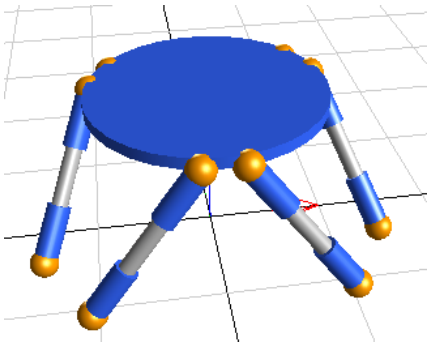


图 12-4　Stewart 模型 2

3. 方便的仿真分析：运动学仿真和动力学仿真都是一次建模可适用于多种输入，模型建立完成后，只需求修改输入参数，即可输出该参数下的仿真结果。

运动学仿真：指定上平台的运动可测出各连杆移动副的伸长量、速度和加速度。用于判断给定运动的性能，常用于轨迹规划性能评定。

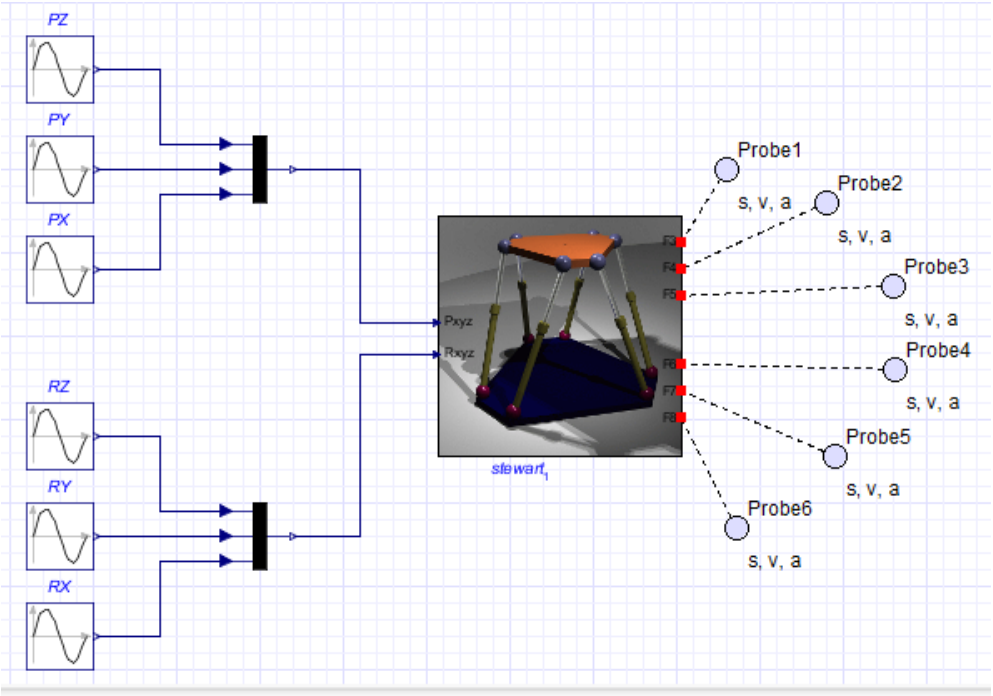


图 12-5　Stewart 运动学仿真模型

动力学仿真：由六个分支作为主动件，驱动上平台运动，由此测出当前运动时各驱动机构所需的驱动力/力矩大小；Stewart 动力学仿真中，由两个结构一样的模型组成，前一个模型为平台驱动进行运动学仿真，得出各分支杆长变化量，直接用于驱动后一个模型，杆长驱动并输出杆长的驱动力。

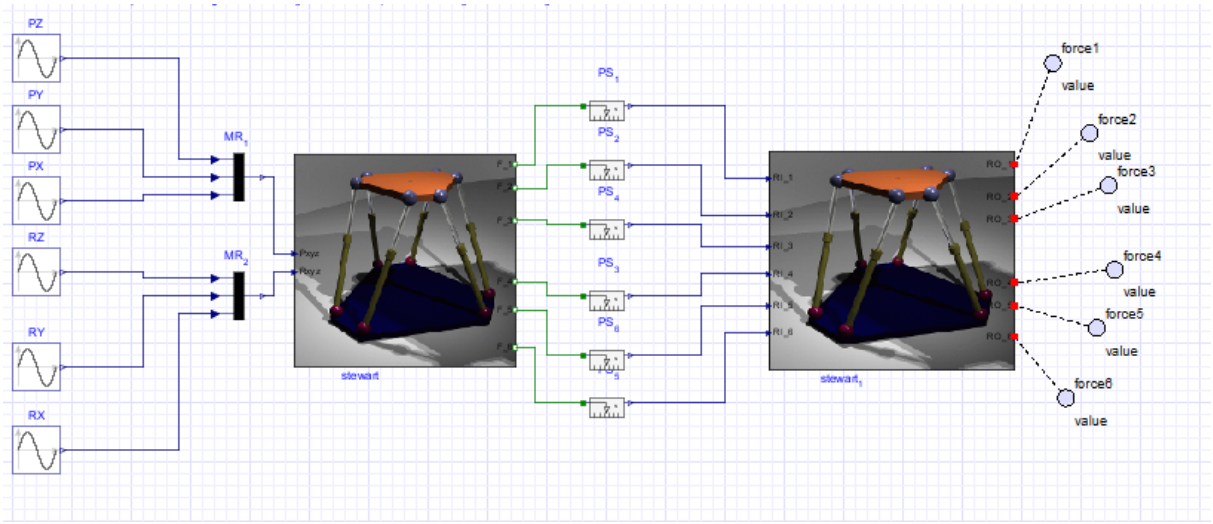


图 12-6　Stewart 动力学仿真模型

4. 开放的模型：可获得模型的运动学和动力学约束方程，无需并联机构模型知识，可实现运动学和动力学控制等，另外可以将 MapleSim 的模型生成被控对象代码导出到

其他仿真控制软件，如生成优化的 Simulink，C++，LabVIEW、FMI 等代码,作为被控对象。

"Analyzing system..."

"Performing constraint analysis..."

"The system has 6 degree(s) of freedom. It is modeled using 24 generalized coordinate(s) coupled by 18 algebraic constraint(s)."

"Peforming a dynamic analysis using an augmented reaction formulation - system variables shown below:"

"vQ", $1 .. 24\ Vector_{column}$ Data Type: anything Storage: rectangular Order: Fortran_order, "vP", $1 .. 24\ Vector_{column}$ Data Type: anything Storage: rectangular Order: Fortran_order, "vPdot", $1 .. 24\ Vector_{column}$ Data Type: anything Storage: rectangular Order: Fortran_order, "vReactions", $1 .. 18\ Vector_{column}$ Data Type: anything Storage: rectangular Order: Fortran_order

"Dynamic analysis complete."

```
> eq3:=vPosCons(3)=0;
```

$$
\begin{aligned}
eq3 :=\ & \cos(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t))\sin(\eta(t))\sin(\xi(t))\cos(\zeta(t))\,YA1 + \cos(LEG1\_U1\_beta(t))\sin(LEG1\_U \\
& - \cos(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t))\sin(\eta(t))\cos(\xi(t))\cos(\zeta(t))\,XA1 - \cos(LEG1\_U1\_beta(t))\sin(LEG1\_ \\
& + \cos(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t))\sin(\xi(t))\sin(\zeta(t))\,XA1 - \cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(\\
& + \cos(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t))\cos(\xi(t))\sin(\zeta(t))\,YA1 - \cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(\\
& + \cos(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t))\cos(\eta(t))\cos(\zeta(t))\,ZA1 + \cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha \\
& - \sin(LEG1\_U1\_beta(t))\sin(\xi(t))\cos(\eta(t))\,YA1 + \sin(LEG1\_U1\_beta(t))\cos(\xi(t))\cos(\eta(t))\,XA1 - y(t)\cos(LEG1\_U1\_ \\
& + z(t)\cos(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t)) + \sin(LEG1\_U1\_beta(t))\sin(\eta(t))\,ZA1 - ZB1\cos(LEG1\_U1\_beta(\\
& + YB1\cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t)) + x(t)\sin(LEG1\_U1\_beta(t)) - XB1\sin(LEG1\_U1\_beta(t)) - LEG
\end{aligned}
$$

```
> eq1:=subs(zeta(t)=A(t),eta(t)=B(t),xi(t)=C(t),eq1);
> eq2:=subs(zeta(t)=A(t),eta(t)=B(t),xi(t)=C(t),eq2);
> eq3:=subs(zeta(t)=A(t),eta(t)=B(t),xi(t)=C(t),eq3);
```

图 12-7　获取 Stewart 模型运动学和动力学方程

5.Stewart 运动控制控制：指定平台期望轨迹，通过模型逆运动学方程模块，计算出实现该运动时各连杆的伸长量，使平台按着期望轨迹运行，以模型方程为基础实现运动控制。

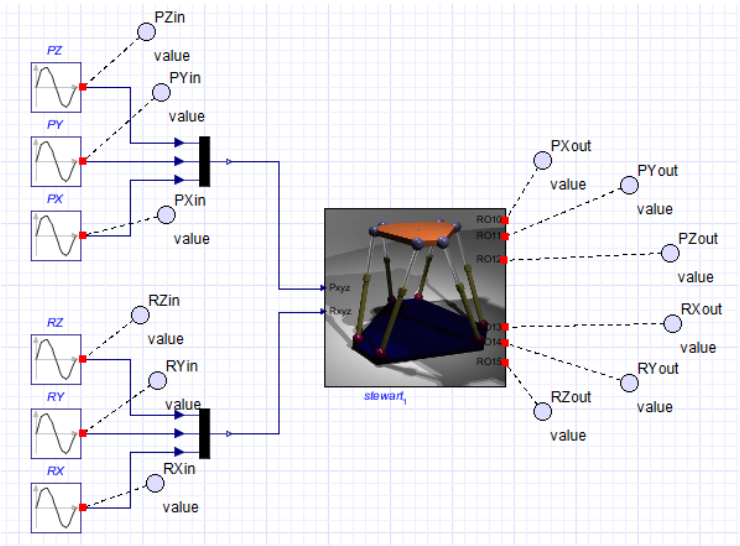


图 12-8　Stewart 运动控制模型

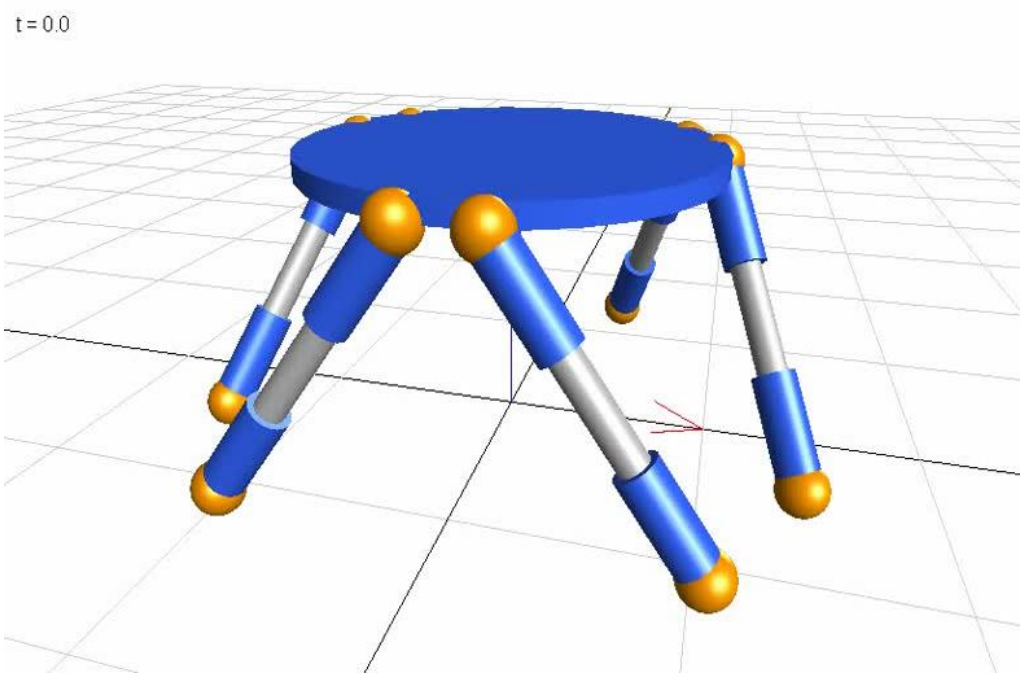


图 12-9　Stewart 运动控制模型动画

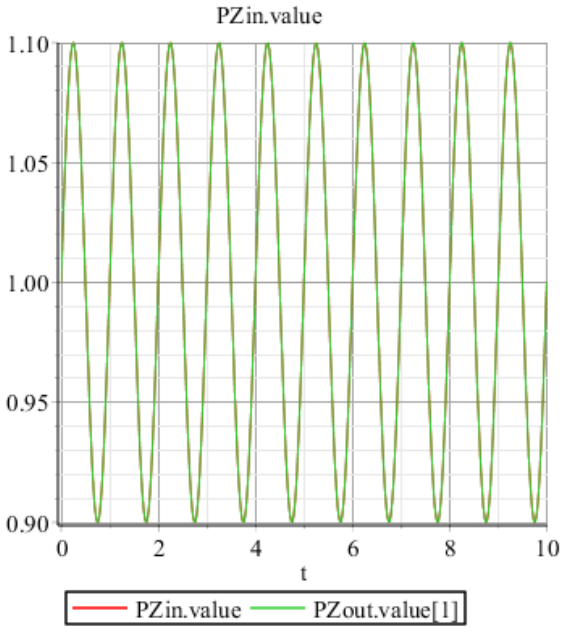


图 12-10　Stewart 仿真结果

12.1 练习 1：Stewart 平台建模

概要

本节将指导用户通过 MapleSim 搭建具有可参数化的 Stewart 模型，可适用于不同

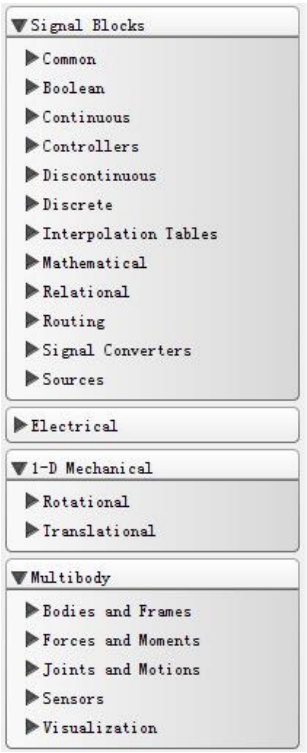
结构大小的 Stewart 模型，提高模型的重用性。

主要内容有：

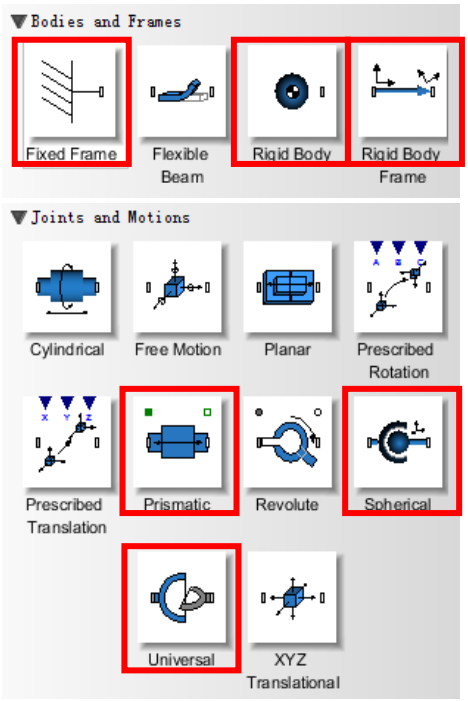
1.分支创建：Stewart 由六个结构相同的分支和上下平台组成，可以创建一个分支后共享成六个分支，提高建模速度。

2.设定参数：参数化的 Stewart 模型主要体现在各坐标点及初始高度由参数 XA1，YA1，ZA1，XB1，YB1，ZB1 和 H 指定。

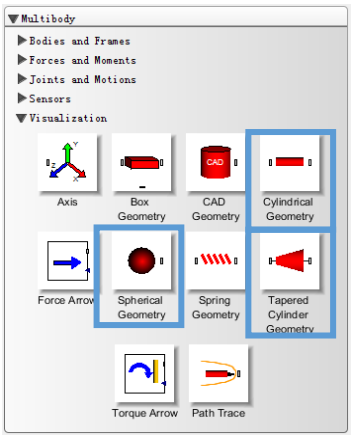
该实例将用到的元件：



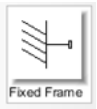
(A)主要的元件库



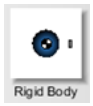
(B) 多体元件库多体元件和运动副



(C)多体元件库可视化元件

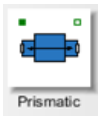


用于定义多体系统的固定端



用于定义多体元件的质心 具有质量和惯量等属性

用于定义多体元件的相对坐标值 类似于连杆

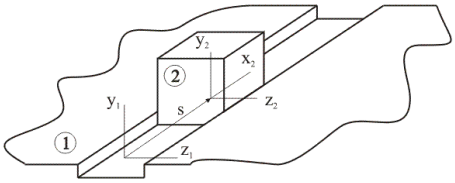


对应于工程上的移动副

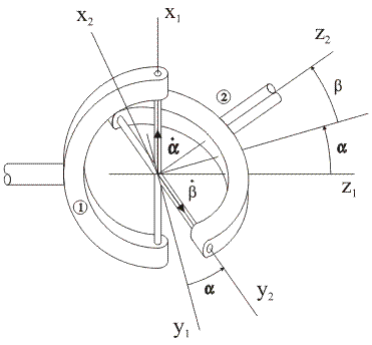
对应于工程上的球副

对应于工程上的万向副

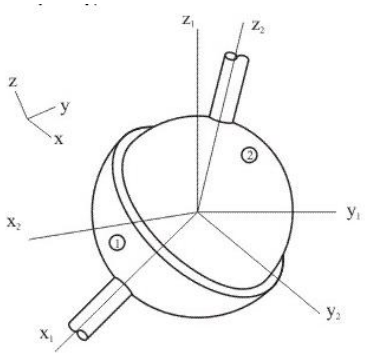
(D)元件解析



移动副 P



万向副（虎克铰）U



球副 S

(E)各个运动副的物理模型

图 12-11　实例中需要的元件

模型下载地址：http://www.maplesoft.com.cn/book/12.1.zip

12.1.1 搭建支脚

1.将分支的所有元件添加到 MapleSim 工作窗口中，采用 U-P-S 的分支结构，如下图：

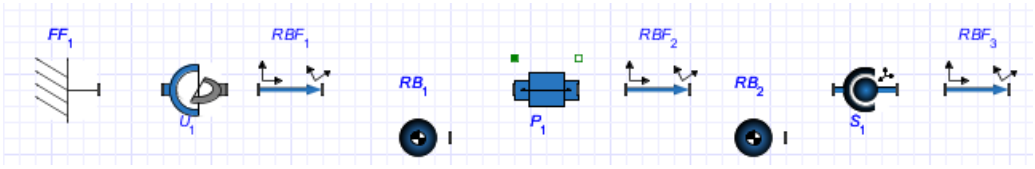


图 12-13　UPS 连杆组成元件

2.使用右击旋转元件，或快捷键 Ctrl+R 顺时针旋转, Ctrl+L 逆时针旋转 ,Ctrl+H 水平翻转,Ctrl+F 垂直翻转。旋转元件如下图：

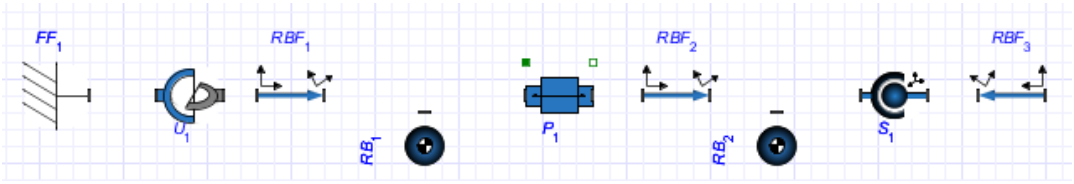


图 12-14　UPS 连杆元件布置

3.按顺序从左到右依次连线，如下图：

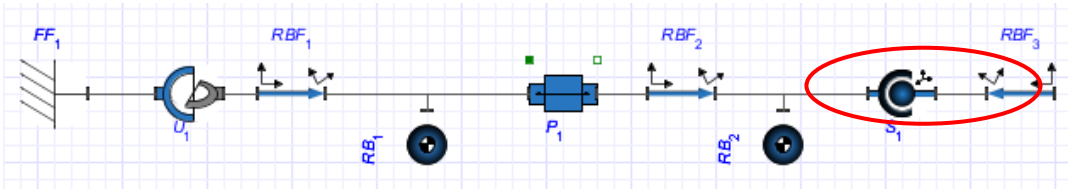
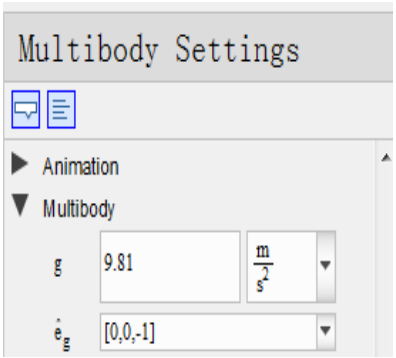


图 12-15　连接连杆元件

12.1.2 参数配置

1.设置重力方向为-Z 方向和大小为 9.81



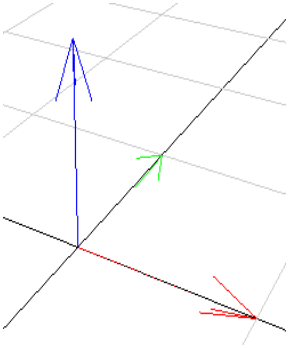


图 12-16　设置重力方向和大小

2.设置分支的各项参数

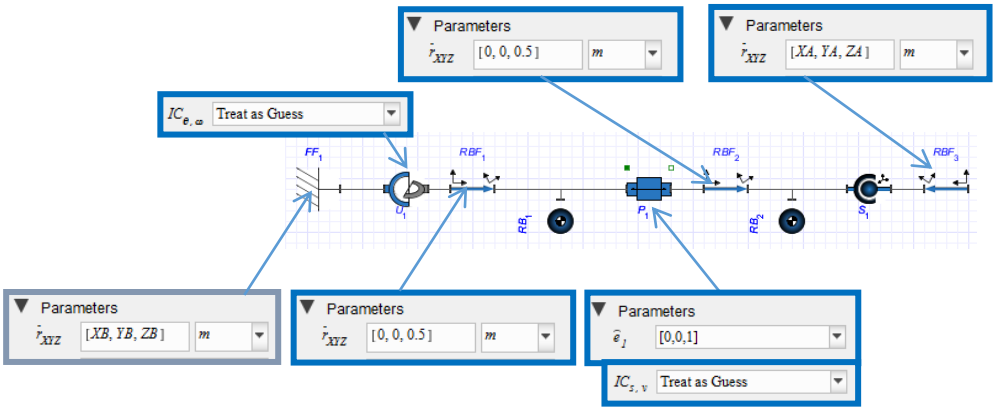


图 12-17　修改连杆元件参数

3.各项参数的意义解析

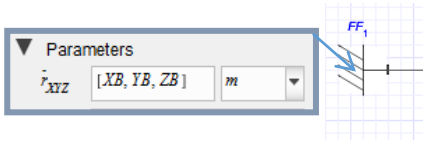


图 12-18　下平台坐标点 XB,YB,ZB

定义上平台的固定端的坐标，采用参数 XB,YB,ZB 定义。

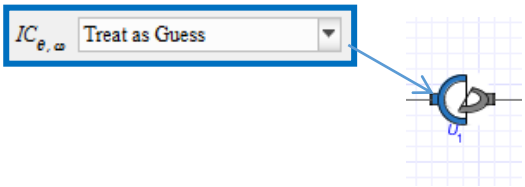


图 12-19　设定 U 副初始值类型

定义 U 副的初始值类型为 Treat as Guess，即系统寻找最优初始参数，同时指定模型方程生成的输出变量。

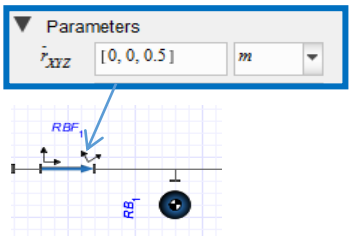


图 12-20　指定连杆坐标

定义质心到 U 副的相对坐标(0,0,0.5)。

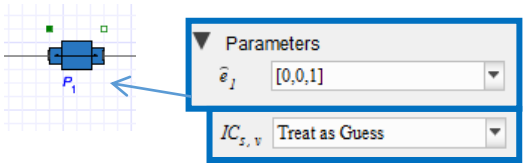


图 12-21　定义 P 副的方向和初始值类型

定义移动副的方向为 Z 方向，类型为 Treat as Guess。

12.1.3 创建子系统

1.全选连杆的模型，右击选择创建子系统项 Create Subsystem，或使用快捷键 Ctrl+G 创建子系统。将子系统命名为 LEG。

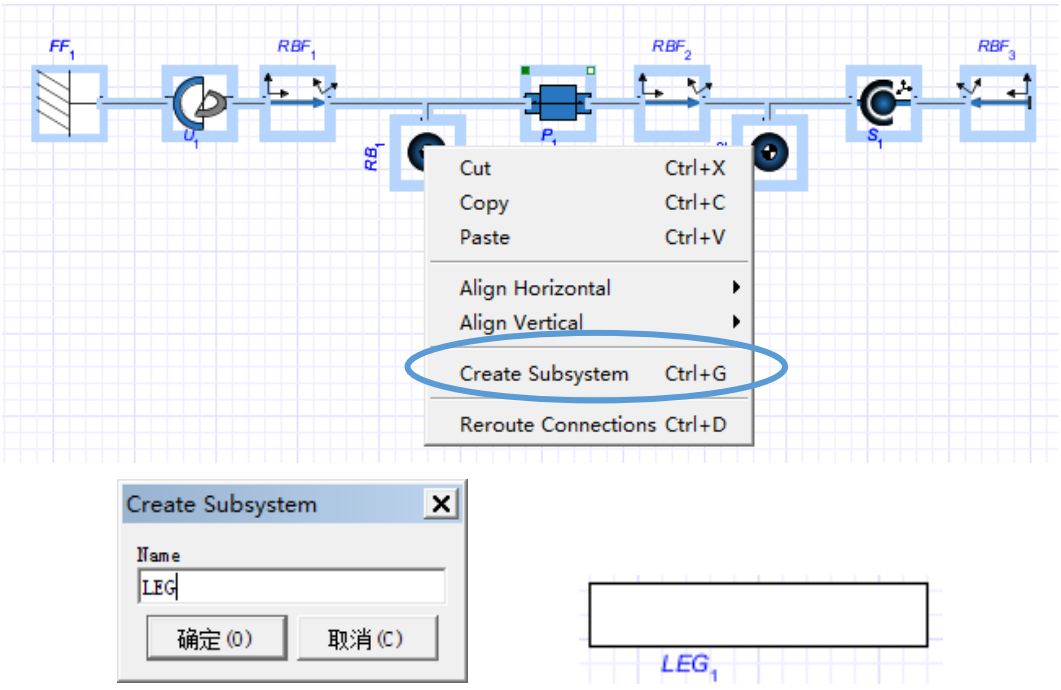


图 12-22　创建连杆子系统 LEG

2.调整子系统的框图大小，接着复制子系统，选择转换为共享子系统（默认设计），共享系统使得所有系统一致，复制到 6 个分支。

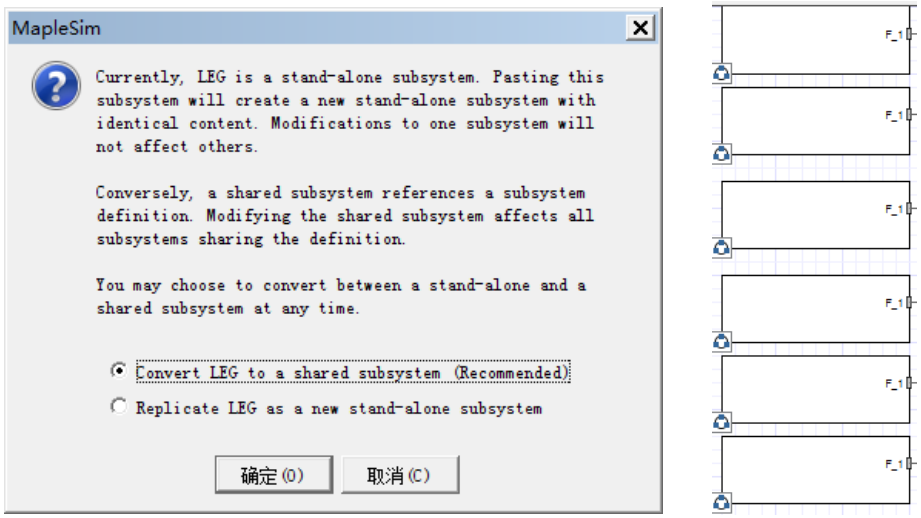


图 12-23　共享连杆子系统

3.双击子系统 LEG 进入子系统，将末端的 RBF_3 连至子系统端口，作为该子系统与其他系统的接口。

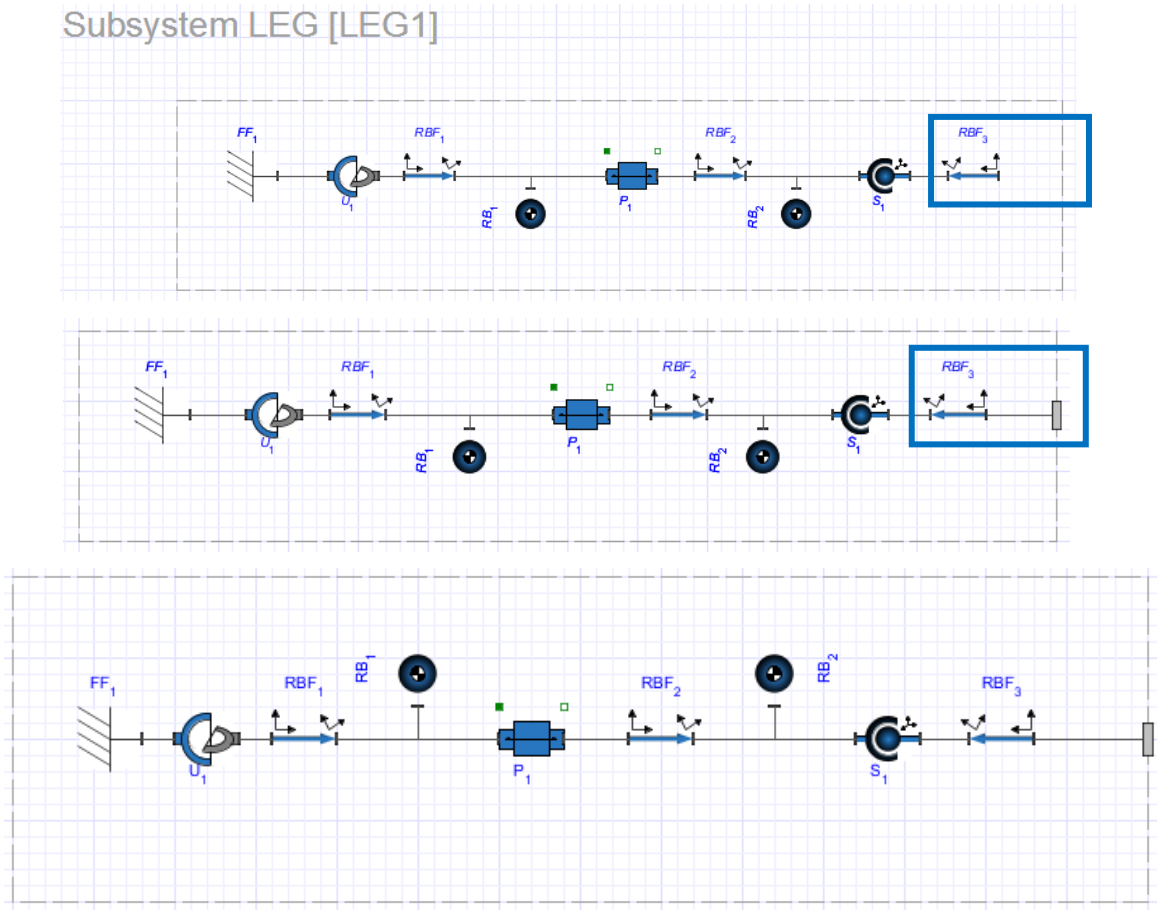


图 12-24　添加连杆子系统端口

4.在 LEG 子系统目录下进入参数配置界面，配置子系统的参数。

图 12-25　参数配置图标

5.输入上下平台的变量 XA,YA,ZA,XB,YB,ZB，上平台用变量 A 表示，下平台用变量 B 表示，X,Y,Z 分别表示在该方向的分量。采用参数建模的形式可提高模型重用性。

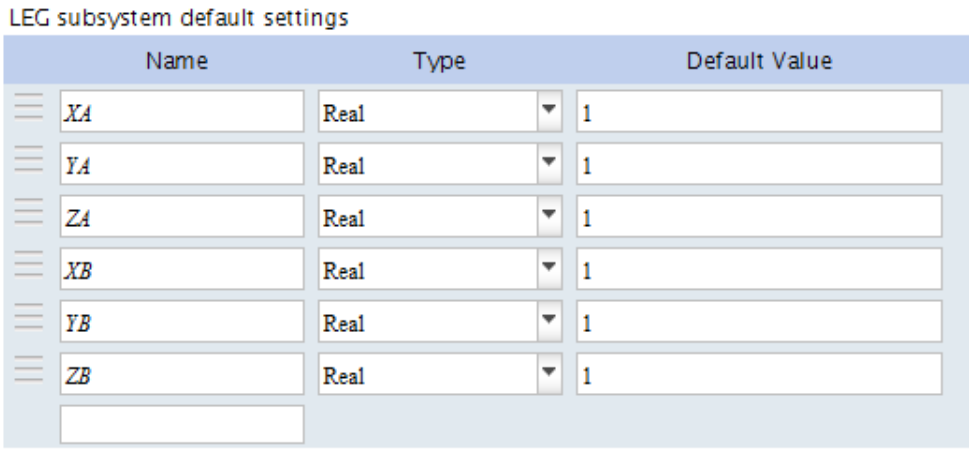


图 12-26　设定 LEG 子系统参数

12.1.4 创建上平台

1.添加 Rigid Body 作为上平台的质心块，并与每个分支连接，如下图所示。

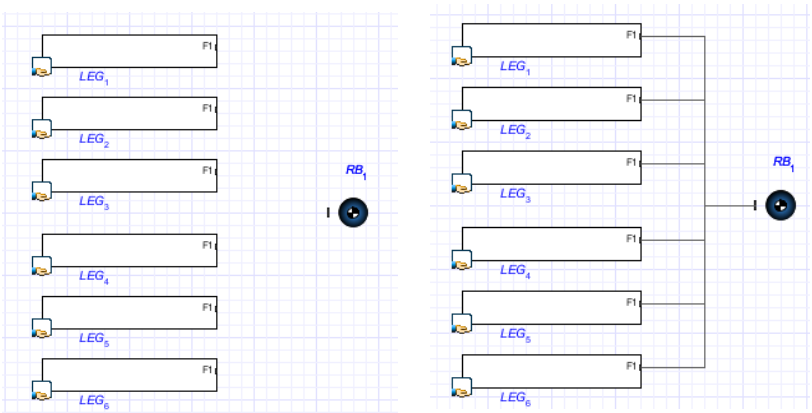


图 12-27　连杆与上平台相连

2.设定上平台的坐标(0,0,H)，并设为强制形式 Strictly Enforce ,使得上平台初始值为(0,0,H)。

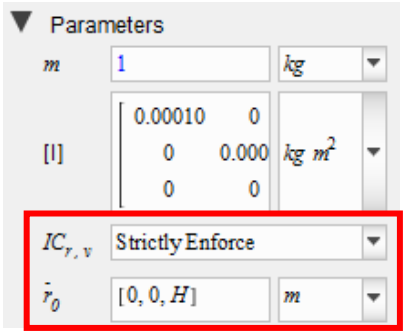


图 12-28　设定上平台初始值坐标和类型

12.1.5 配置模型参数

1.点击参数块配置模型参数,修改参数块的名字为 P_1，进入参数块输入参数名称 XA1，YA1，ZA1，XB1，YB1，ZB1 分别作为 Stewart 平台第一分支的与上下平台相连的坐标点。

图 12-29　添加参数块 P_1

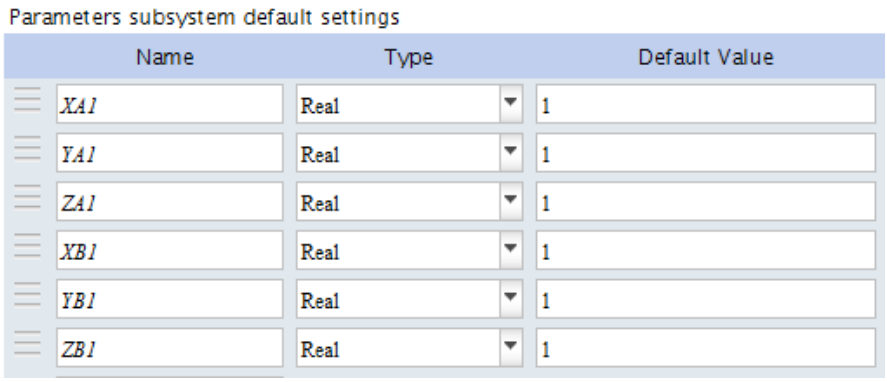


图 12-30　配置参数块变量

2.复制参数快 P_1，选择复制为独立 stand-alone 模式，独立模型可作为独立的元件，复制到 6 个参数模块。

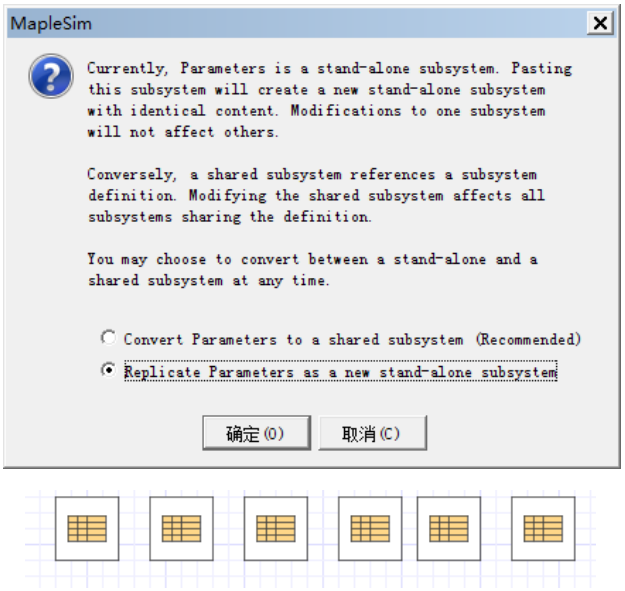


图 12-31　六个独立的参数块

3.分别将 P_2 至 P_6 的参数块的参数名称如下，分别为其他分支的坐标点，参数的值后面再修改。

P_2-------XA2，YA2，ZA2，XB2，YB2，ZB2.

P_3-------XA3，YA3，ZA3，XB3，YB3，ZB3.

P_4-------XA4，YA4，ZA4，XB4，YB4，ZB4.

P_5-------XA5，YA5，ZA5，XB5，YB5，ZB5.

P_6-------XA6，YA6，ZA6，XB6，YB6，ZB6.

请大家都使用这个变量名称，后面的参数设定是基于该变量下进行。

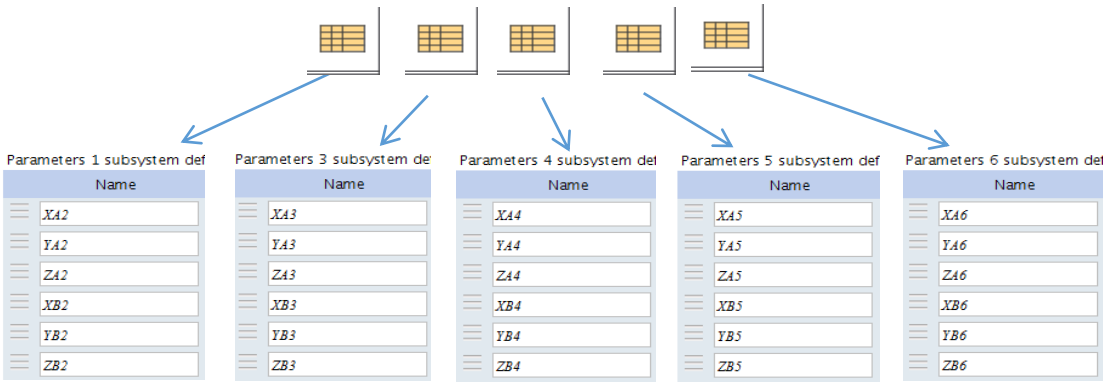
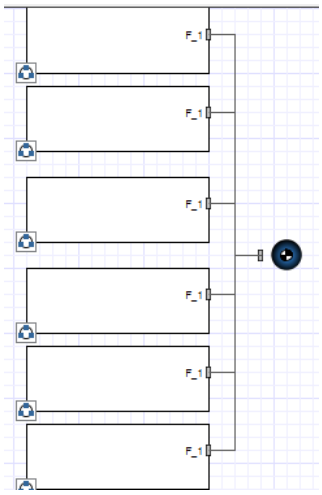


图 12-32　修改参数块变量

4.点击连杆子系统，将各分支的参数设置如下：如 LEG_1 参数设为 XA=XA1,YA=YA1,ZA=ZA1 等。



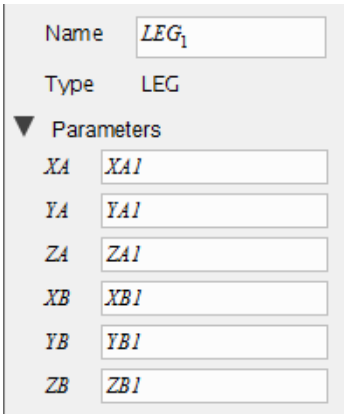


图 12-33　连杆参数传递

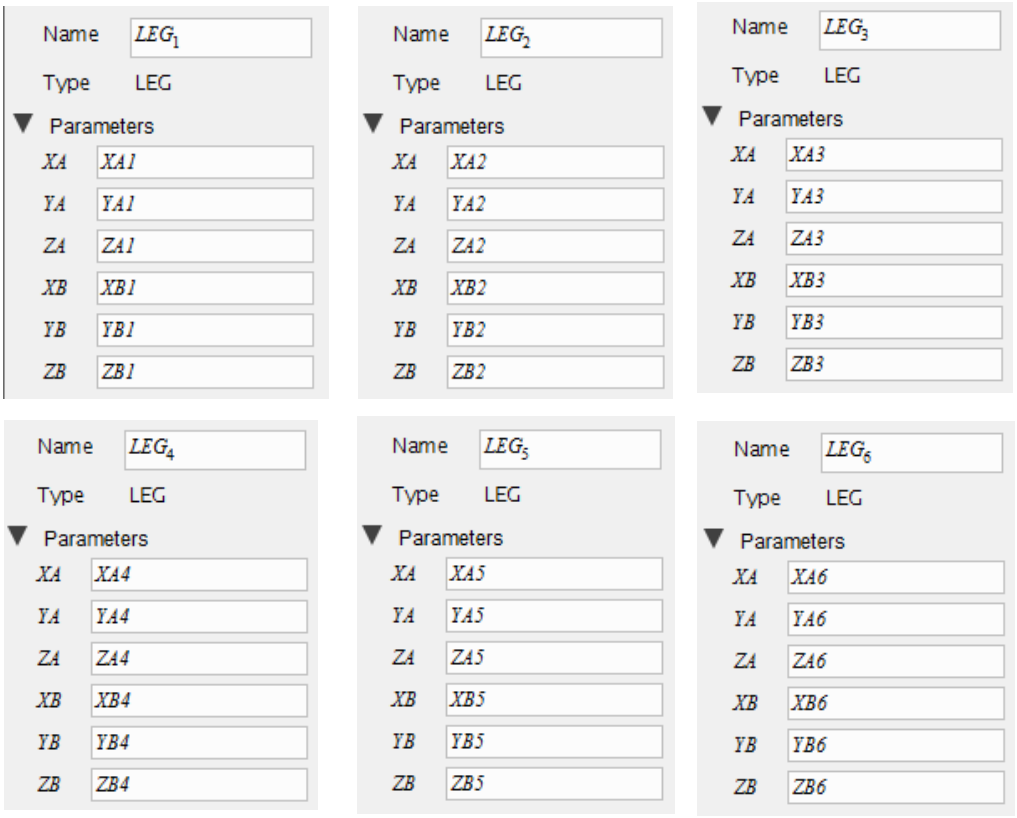


图 12-34　连杆参数传递

5.点击参数块配置模型参数,修改参数块的名字为 platformset ， 进入参数块输入参数名称 H,用于定义上平台的初始高度。

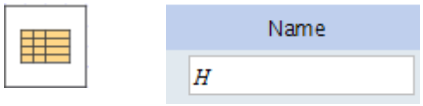


图 12-35　上平台初始值参数

6.检查 Stewart 模型，整体图形如下，重点检查各参数块变量名称是否一一对应。

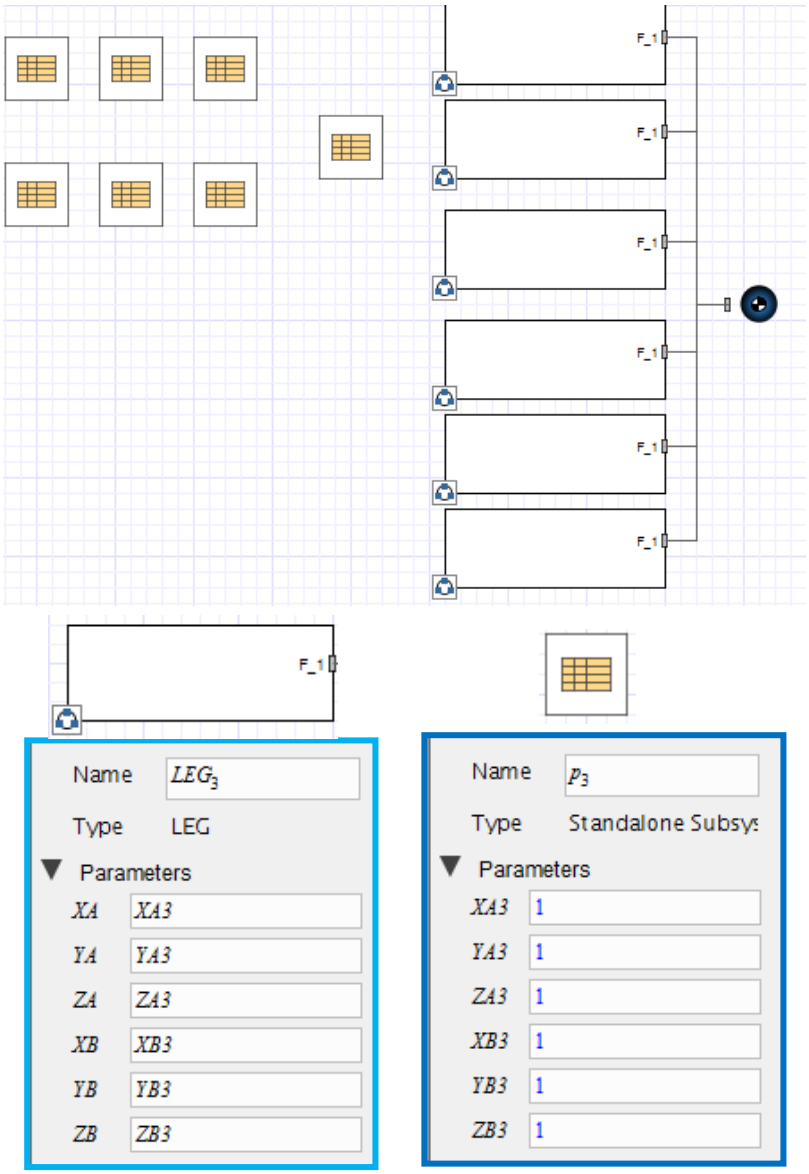


图 12-36　检查模型参数

12.1.6 修改模型参数

1.点击左侧项目面板的 Add Apps and Templates 图标（  ）为 MapleSim 模型文件添加分析模板。

2.点击 Browse 按钮选择 Set paramters (自定义分析模板，位于电脑桌面 Stewart 模型文件夹中)，如下图所示，点击 Create Attachment 进入 Maple 环境。

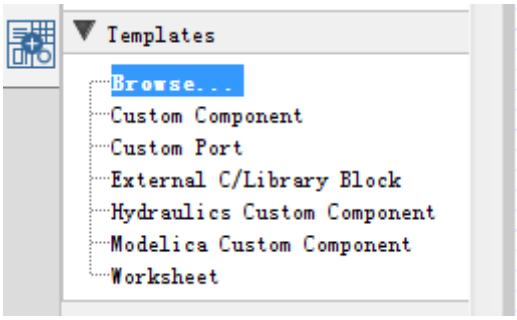
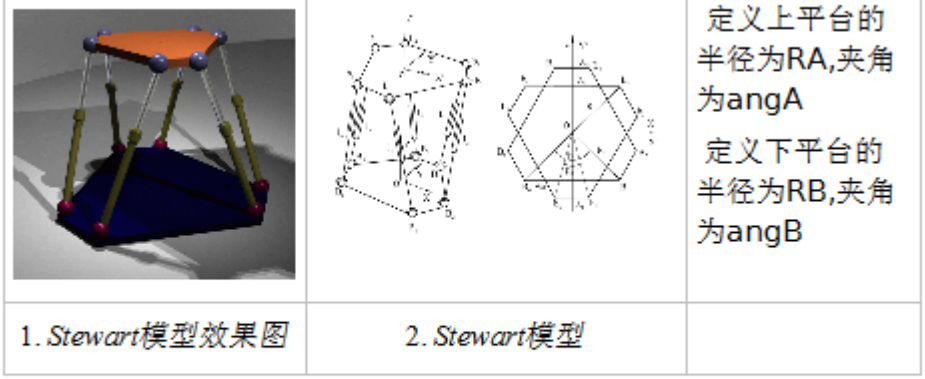


图 12-37　调用自定义分析模板

3.进入 Set paramters 分析模板后的界面如下，通过该模板可以通过程序读取 Stewart 模型上平台半径 RA，下平台半径 RB，上平台夹角 angA，下平台夹角 angB 以及模型初始位置高度 H_0，来确定 Stewart 模型各关节点的坐标值。

```
> restart:###清空系统的内存
> A := MapleSim:-LinkModel():###读取当前的maplesim模型
> A:-GetParameters(allparams);###获取当前的参数
```




```
> RA:=1:RB:=1: angA:=0:angB:=0:H0:=1:###定义上下平台的半径RA,RB,两点间的夹角,定义上平台的初始高度H=1
```

图 12-38　自定义分析模板界面

4.先确保 Stewart 模型中的参数名称定义为上平台 XA,YA,ZA,下平台 XB,YB,ZB 并以 1,2,3,4,5,6 区分六个分支(按顺序定义分支)，程序执行完后，可获得修改后的参数。

```
> A:=GetParameters(allparams);###获取修改后的参数
```

$$\left[XA1=0, XB1=\frac{1}{2}\sqrt{3}, YA1=-1, YB1=-\frac{1}{2}, ZA1=0, ZB1=0, XA2=\frac{1}{2}\sqrt{3}, XB2=\frac{1}{2}\sqrt{3}, YA2=\frac{1}{2}, YB2=-\frac{1}{2}, ZA2=0, ZB2=0, XA3=\frac{1}{2}\sqrt{3}, XB3=0, YA3=\frac{1}{2}, YB3=1, ZA3=0, ZB3=0, XA4=-\frac{1}{2}\sqrt{3}, XB4=0, YA4=\frac{1}{2}, YB4=1, ZA4=0, ZB4=0, XA5=-\frac{1}{2}\sqrt{3}, XB5=-\frac{1}{2}\sqrt{3}, YA5=\frac{1}{2}, YB5=-\frac{1}{2}, ZA5=0, ZB5=0, XA6=0, XB6=-\frac{1}{2}\sqrt{3}, YA6=-1, YB6=-\frac{1}{2}, ZA6=0, ZB6=0\right] \quad (4)$$

图 12-39　执行程序得到修改后参数

5.回到 Maplesim 模型，把获取到的参数修改到 Parameters 中。

图 12-40　最新参数显示

12.1.7 运行模型

1.点击 Maplesim 的运行按钮，蓝色播放图标，图标转成停止按钮表示正在仿真求解.

2.仿真结束，在图标窗口，点击，查看 Stewart 三维模型，如下图所示。

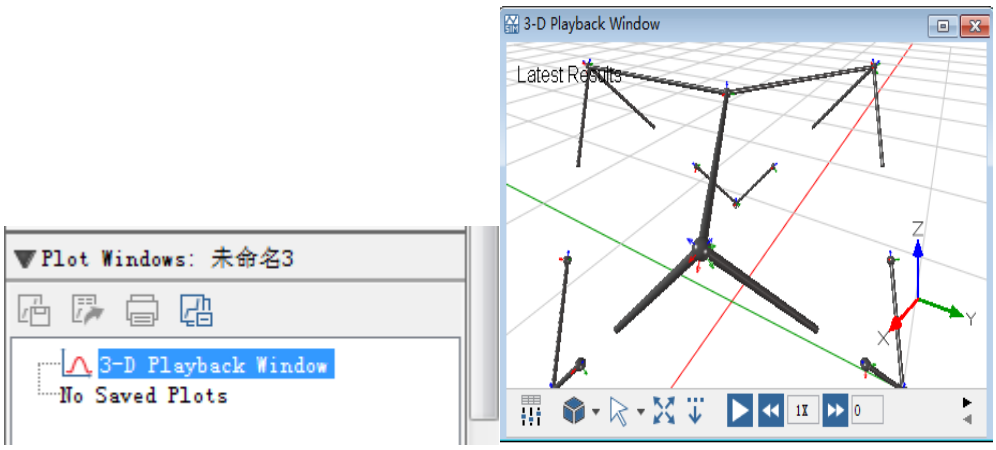


图 12-41　Stewart 三维模型

3.保存模型为 stewartmodel。

12.1.8 模型可视化

1.在 Multibody 的 Visualization (多体可视化)目录下，选择 Cylindrical Geometry (圆柱体)用于表示各分支的形状，选择 Spherical Geometry(球体)用于表示上下平台的球副和万向副，选择 Tapered Cylinder Geometry(锥形体)用于表示上平台。

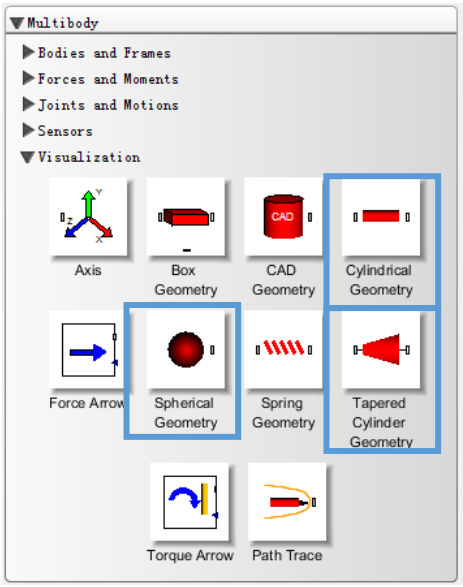


图 12-42　多体可视化元件库

2.在 Multibody 的 Visualization (多体可视化)目录下，拉出下图的可视化元件

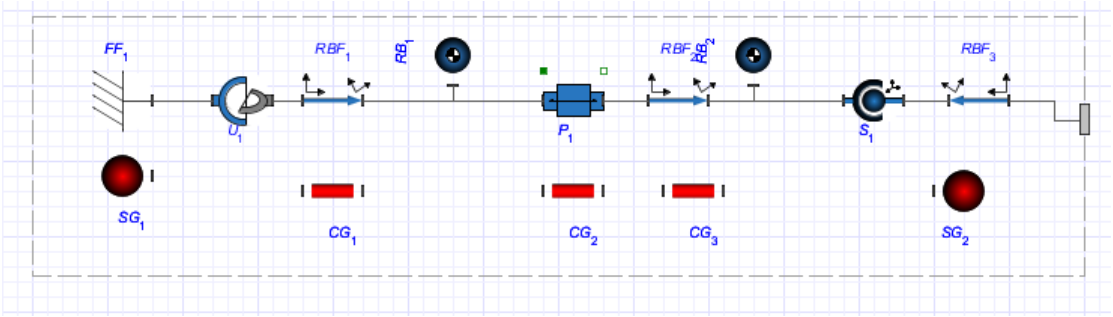


图 12-43　可视化元件放置到到分支子系统内

3.按顺序将可视化元件连接到分支中，最终连线图如下图所示。

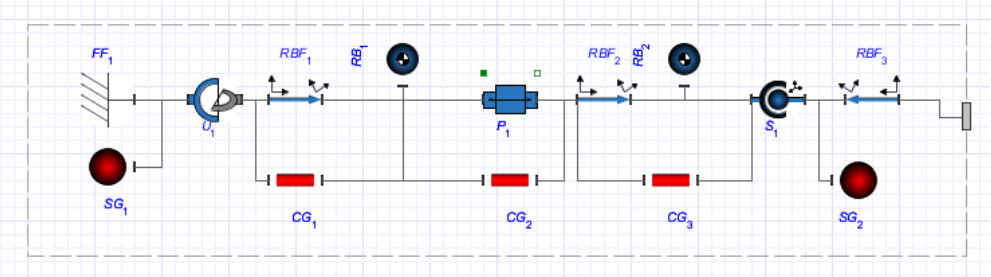


图 12-44　可视化元件连接到分支上

4.具体的连接步骤。

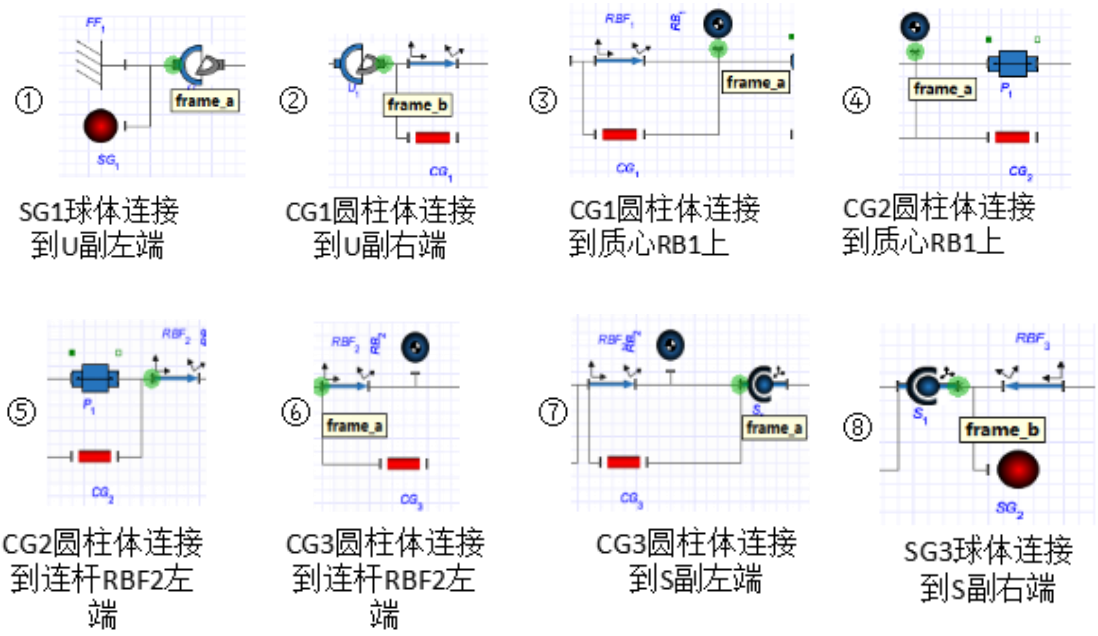


图 12-45　连线步骤

5.修改可视化元件的颜色和半径大小，按如下图设定。

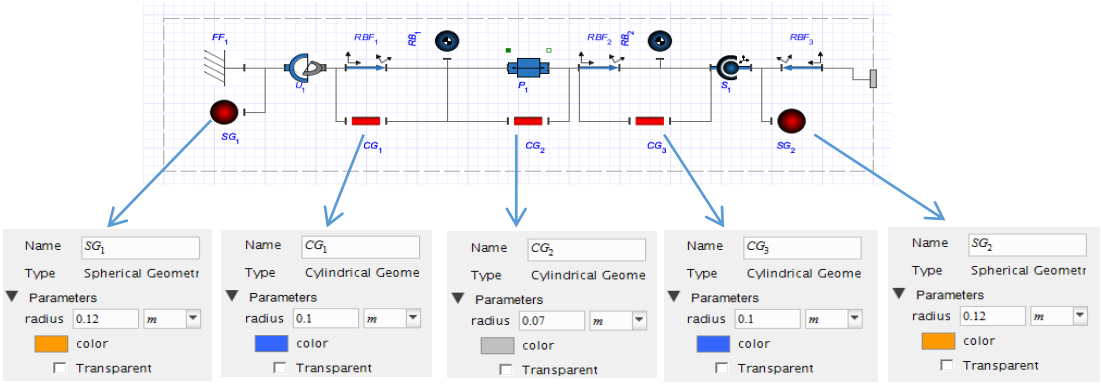


图 12-46　修改可视化元件的颜色和大小

6.配置上平台的可视化模型。

拉出一个 Rigid Body Frame 用于表示上平台的厚度，设定 Rigid Body Frame 坐标参数为(0,0,0.1)，指定上平台厚度为 0.1m。

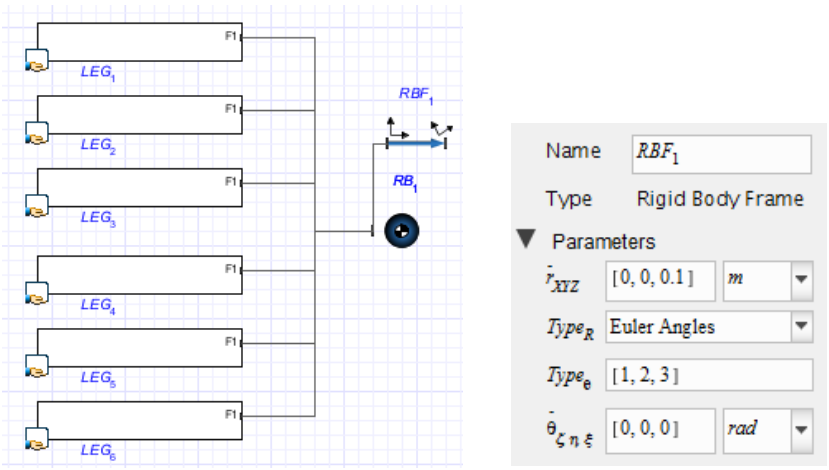


图 12-47　指定上平台的厚度

拉出锥形体 TCG_1 并与 RBF_1 连接，设定锥形体 TCG_1 上下半径和颜色

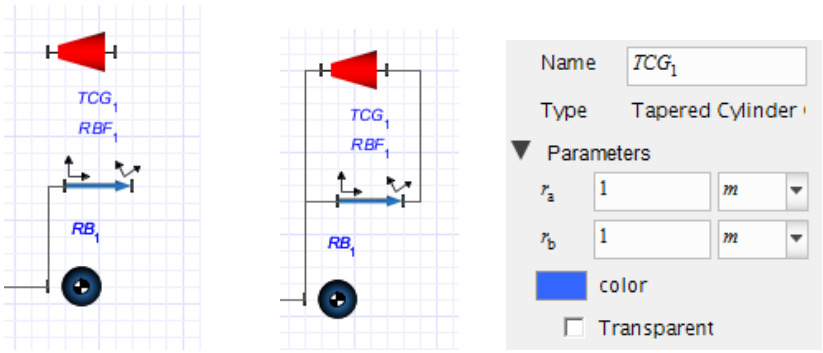


图 12-48　修改上平台可视化的半径

7.再次运行模型，切换到三维显示模式，将内含的几何体隐藏，点击如下框选的按钮，可查看最终的 Stewart 可视化效果。另外可视化模型可以使用 CAD Geometry 调用其他三维软件建立的三维实体模型。

（A）隐藏内含几何体

(B)CAD 三维模型

图 12-49　模型显示方面的修改

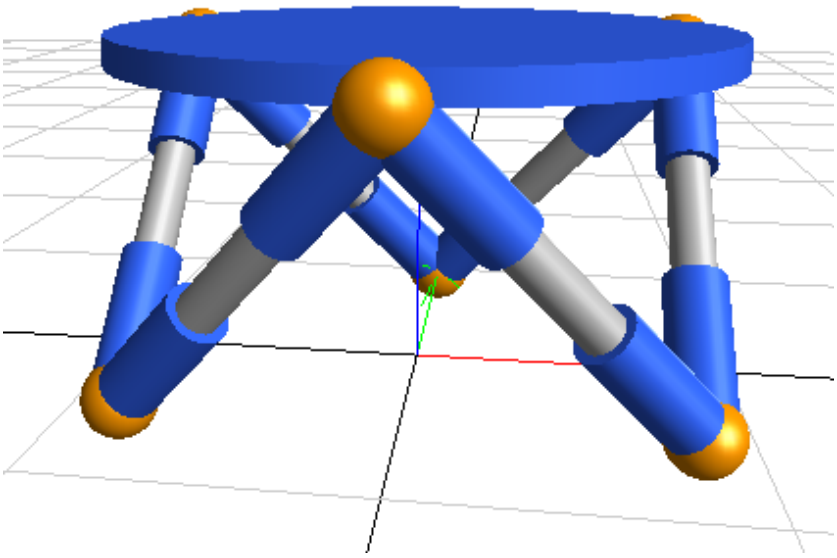


图 12-50　Stewart 模型效果图

12.2 练习 2：Stewart 平台运动学仿真

概要

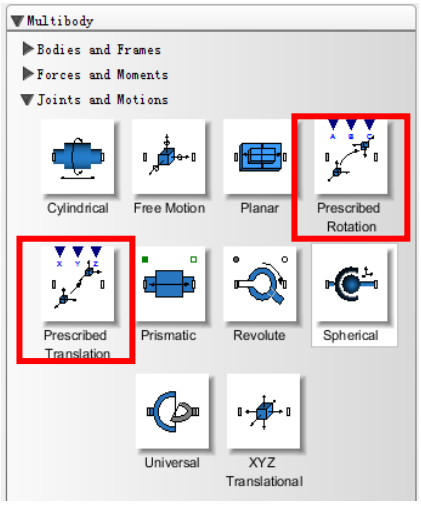
Stewart 的运动分析包括位置分析、速度分析和加速度分析等三部分。运动分析包括两种问题，即运动分析的正解和运动分析的反解。已知输入参数求输出参数是机构的运动学正解；反之已知输出参数求输入参数是机构的运动学反解。

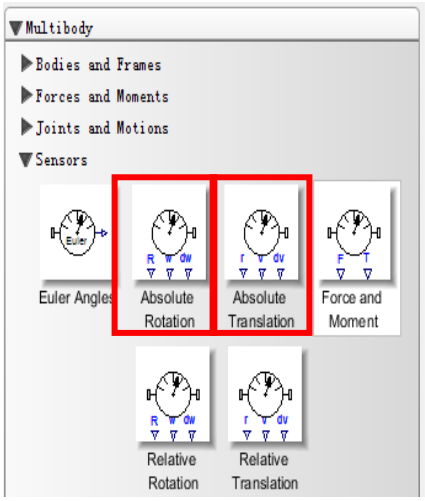
Stewart 运动学仿真为已知上平台的运动，求六个分支的位置、速度、加速度，为运动学反解的范畴。主要内容有：

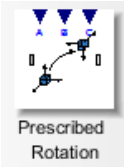
1.给定上平台输入：上平台作为主动件，通过 Absolute Translation, Absolute Rotation 指定上平台质心的运动（位置、姿态）。

2.测量连杆输出：通过 Probe 探针测量连杆位置、速度、加速度。

该实例将会用到的元件：

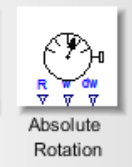




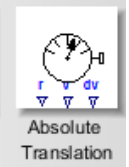


三自由度转动驱动元件

三自由度位移驱动元件



多体系统绝对转动测量元件



多体系统绝对移动位置测量元件

图 12-51　主要使用的元件

12.2.1 另存模型

另存 stewart model 为 stewartkin (Kinematics)，保留 Stewart model 模型用于后续的方程生成。新保存的 Stewartkin 模型的用于运动学仿真。

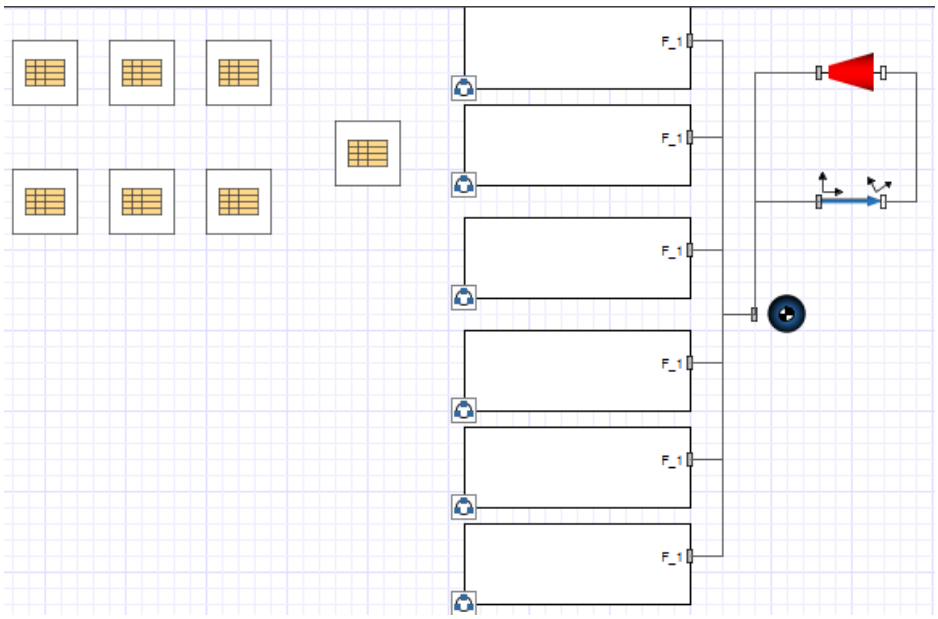


图 12-52　stewartkin 模型

12.2.2 添加上平台驱动

1.将上平台封装成一子系统，框选除六个分支外的所有元件，使用快捷键 Ctrl+G 创建子系统，命名为 platform。

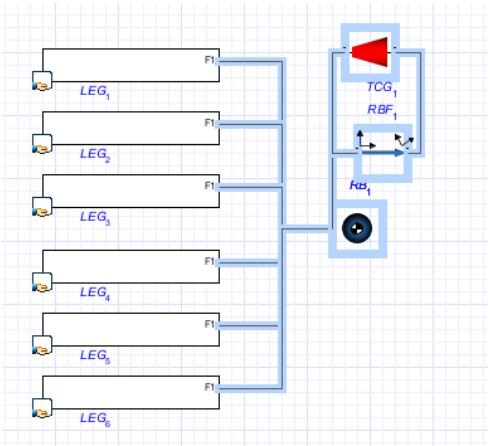


图 12-53　选择上平台组件

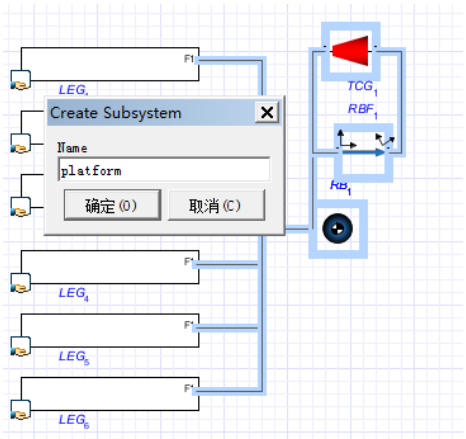


图 12-54　创建子系统

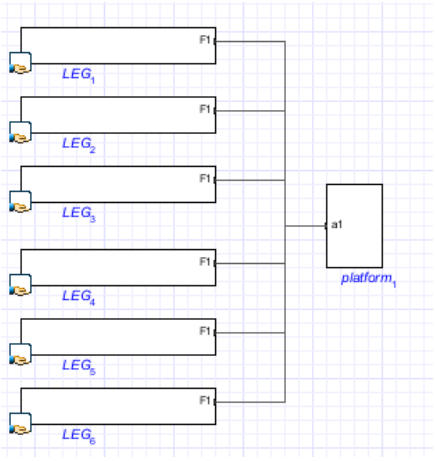


图 12-55　上平台子系统 platform

2.进入上平台 platform 子系统，从多体系统库里添加固定端 Fixed Frame,转动驱动 Prescribed Rotation,移动驱动 Prescribed Rotation,并顺序连接，最后连接到上平台质心块一端。

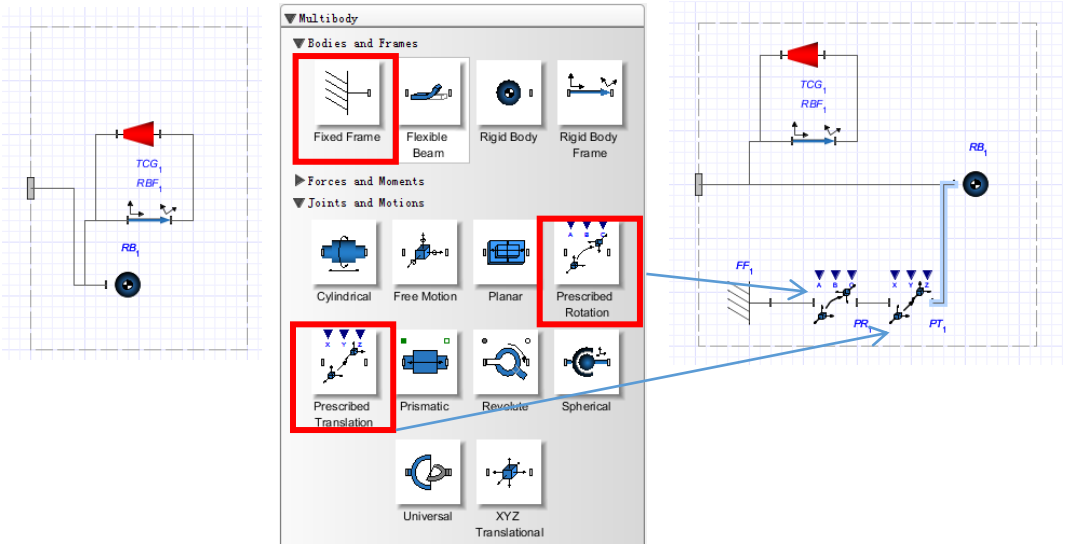


图 12-56　给定上平台驱动

3.拉出两个信号分解元件 Real DeMultiplexer,将转动驱动和移动驱动的输入端口合并成一个，最后将合并后的端口连接到子系统的左侧端口，如下图所示。

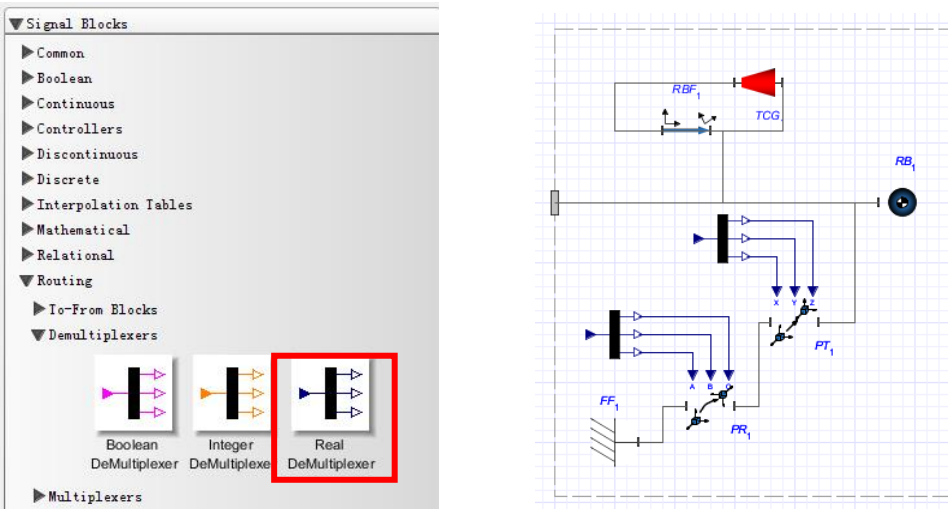


图 12-57　连接信号分解元件

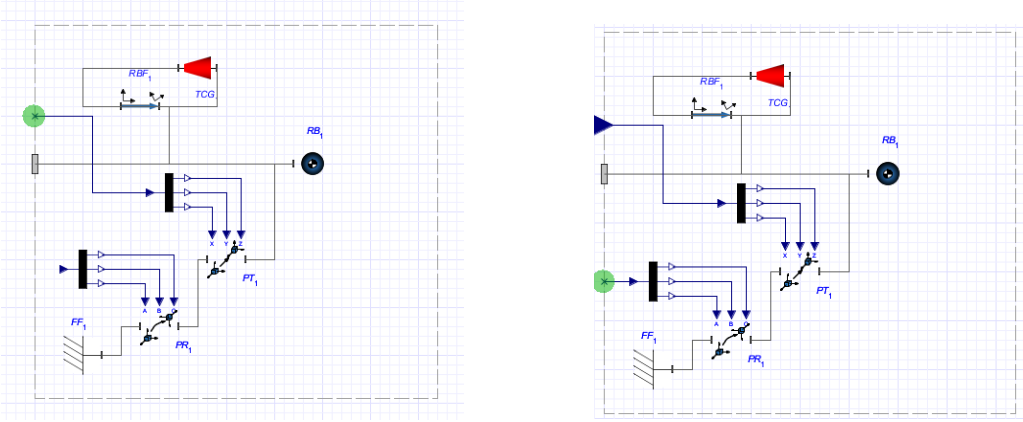


图 12-58　连接上平台驱动端口

4.用鼠标拖动分别框选移动驱动端口，将名称修改为 PIN,同样的将转动驱动端口的名称修改为 RIN。切换到主系统目录下，可以看到 platform 的端口有 PIN,RIN 两个端口。

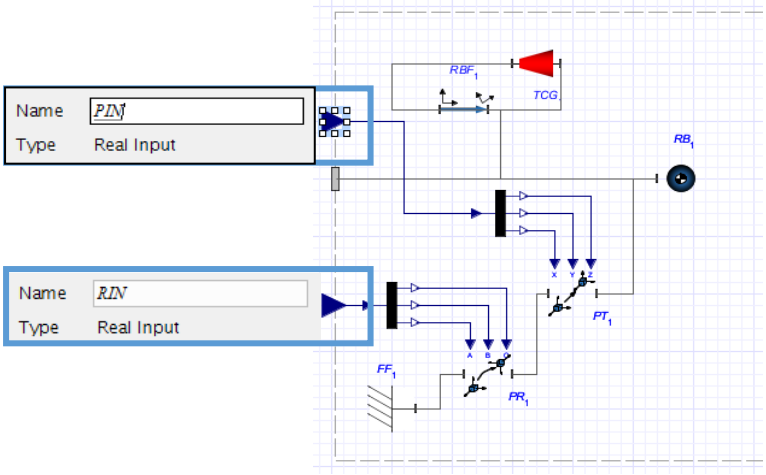


图 12-59　修改端口名称 PIN ,RIN

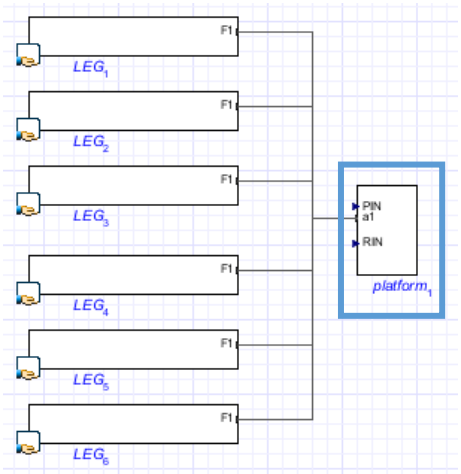


图 12-60　platform 子系统上显示端口名称

5.修改上平台的 Fixed Frame 的坐标为(0,0,H),使得驱动的信号以上平台质心为中心。

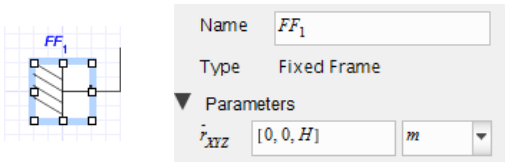


图 12-61　指定驱动输入的初始位置

12.2.3 给定上平台输入

1.将六个分支和上平台组成一个子系统，命名为 stewart。

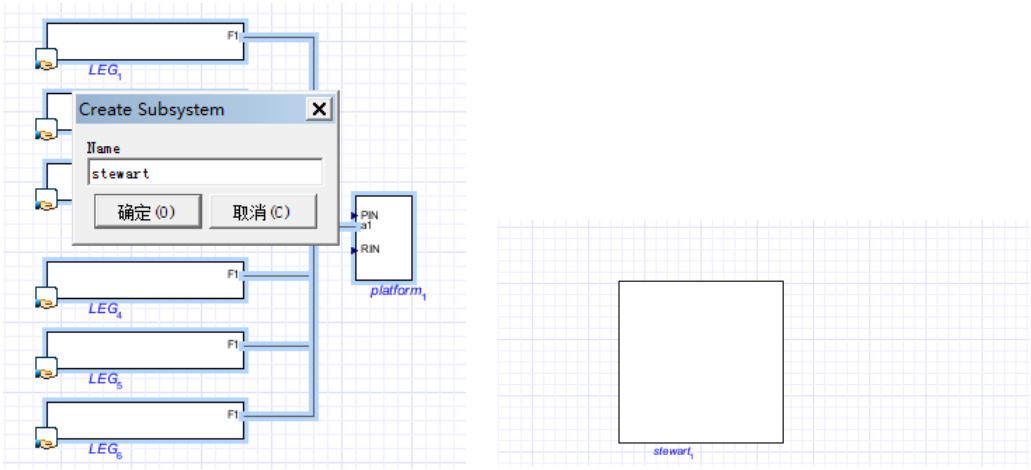


图 12-62　定义整个模型为 stewart 子系统

2.进入 stewart 子系统，将上平台的输入端口 PIN,RIN 引出到该子系统的左侧，采用同样的操作将端口分别命名为 Pxyz,Rxyz。

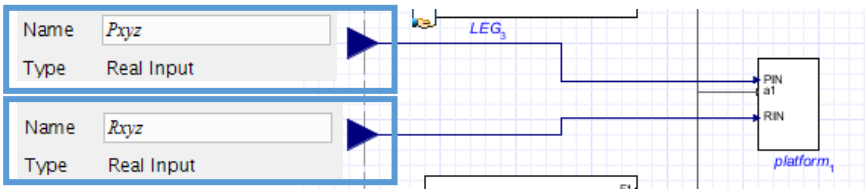


图 12-63　创建 stewart 子系统端口 Pxyz,Rxyz

3.拉出两个合并信号元件 Real Multiplexer,并分别与 stewart 的两个端口连接。接着拉出六个正弦信号依次连接到合并信号元件，作为模型的输入信号。

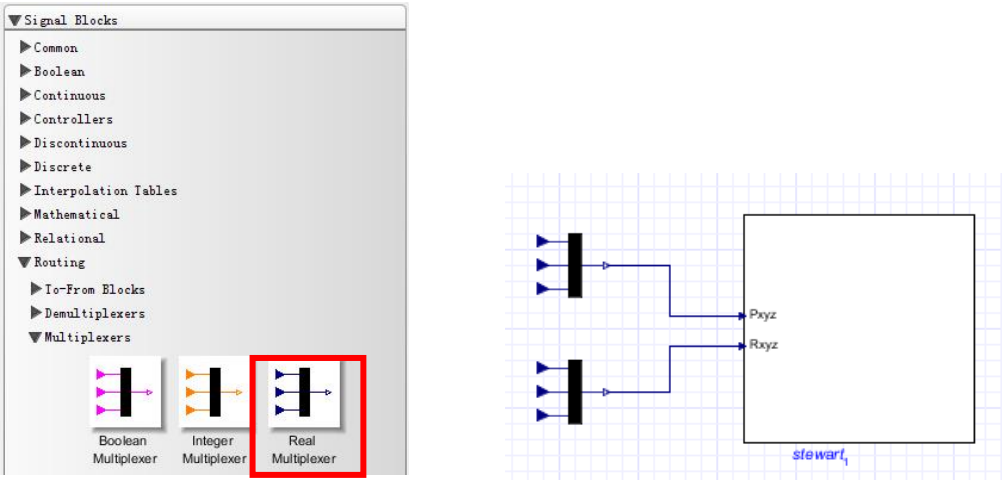


图 12-64　连接信号合成元件到 Stewart 子系统

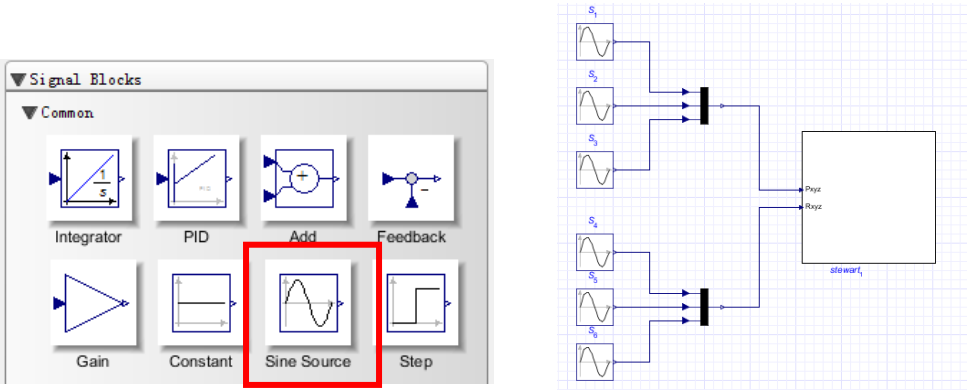


图 12-65　使用 Sine 信号作为平台输入

4.修改六个正弦信号元件的名称分别为 PZ,PY,PX,RZ,RY,YX 便于识别输入信号源，修改输入信号的大小如下(先单个自由度后多自由度复合运动)。

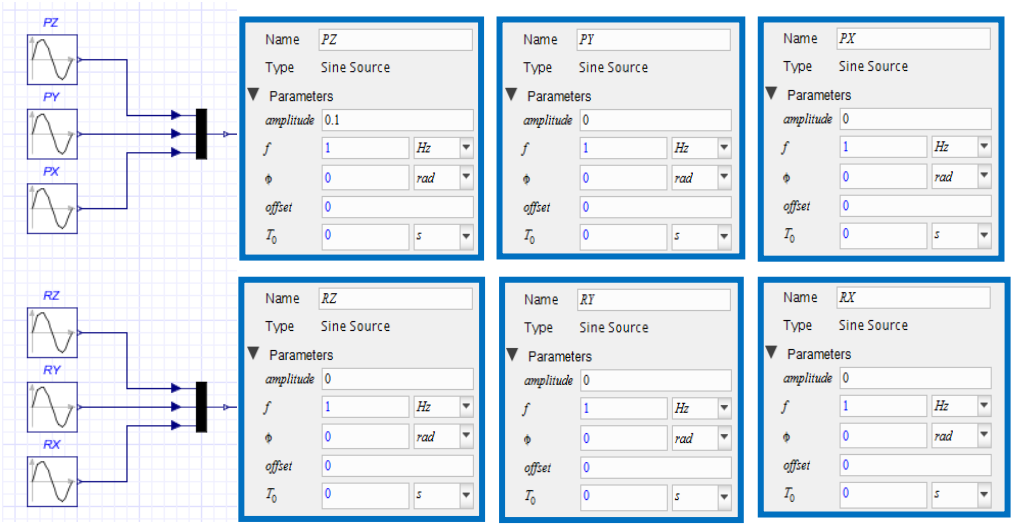


图 12-66　指定输入信号的频率和幅值

12.2.4 测量杆长输出

1.进入连杆子系统 LEG,将移动副 P_1 的右侧端口引出，先连接到子系统 LEG 的虚线边框，再连接到子系统 stewart 的虚线边框，如下图所示。

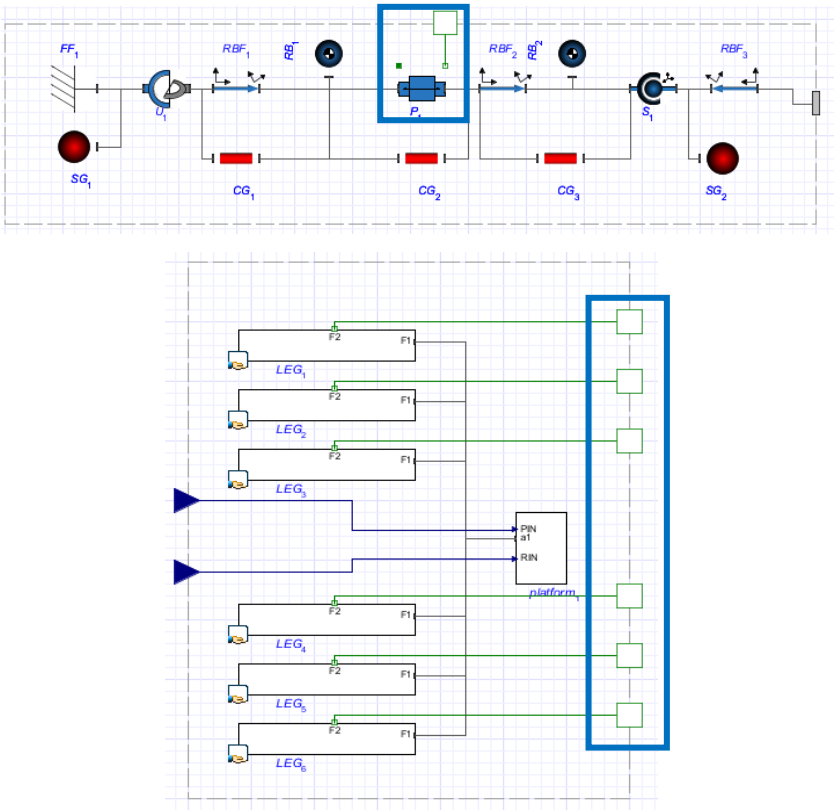


图 12-67　引出杆长输出端口

2.将探针 连接到移动副的端口，可测量仿真后连杆的位置值，杆长速度值，杆长加速度值。可以增加六个探针将所有杆长值输出。

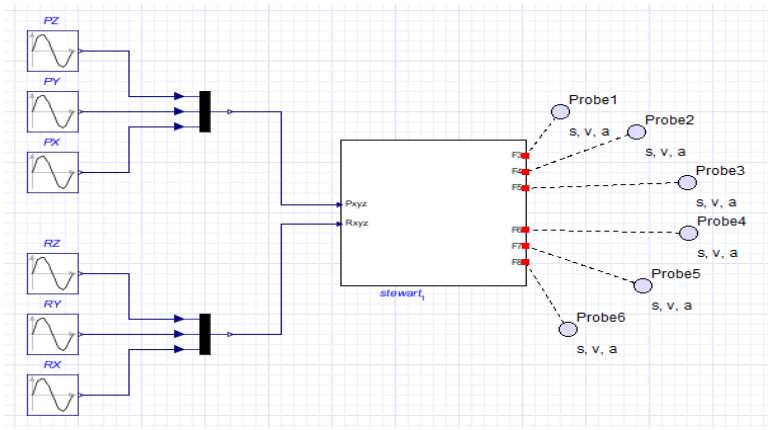
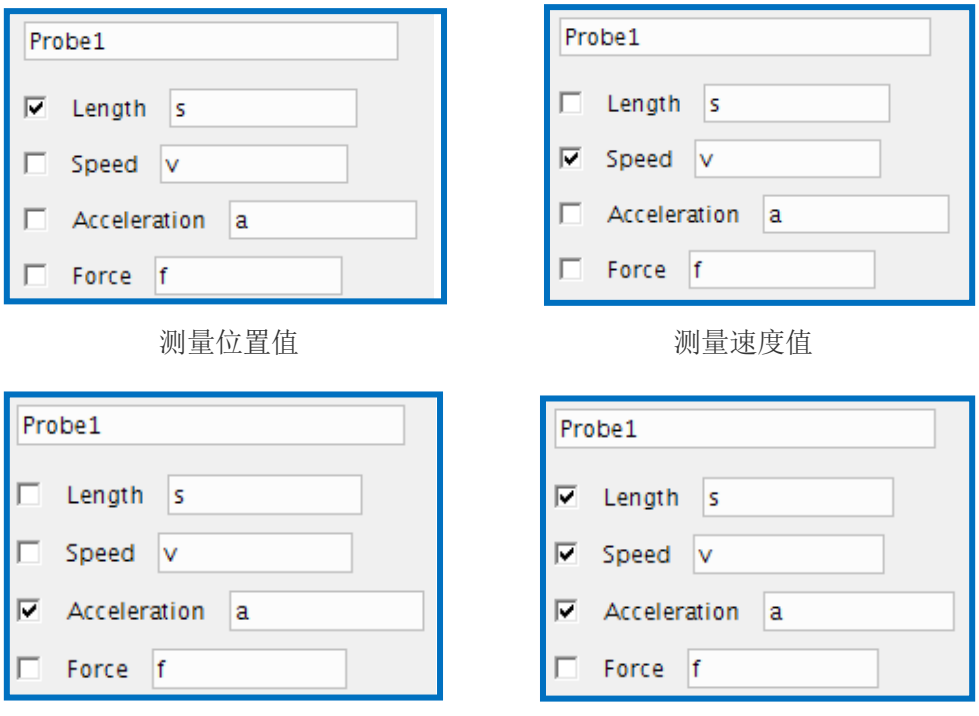


图 12-68 杆长输出测量



测量位置值　　测量速度值

测量加速度值　　三者同时测量

图 12-69　探针输出选择

12.2.5 修改子系统外观

1.进入 stewart 子系统，点击 Icon 图标进入外观配置环境，点击子系统框图接着点击填充选项，通过浏览选择 stewart.gif 的图片(位于培训模型文件)。

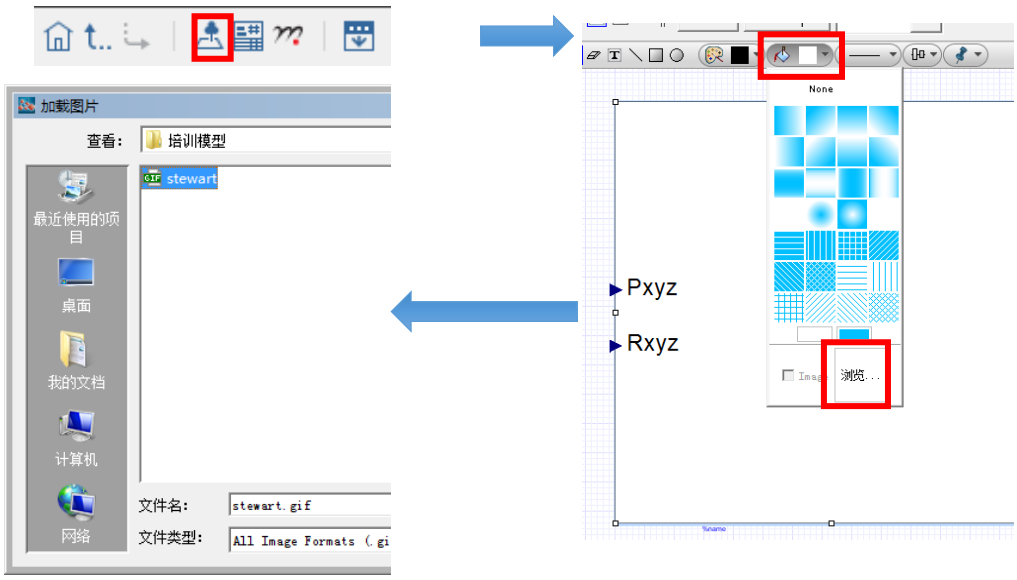


图 12-70　修改 Stewart 子程序外观图标

2.添加完图片后子系统的外观如下，可以通过拖拽子系统的边框使子系统外观更美观。

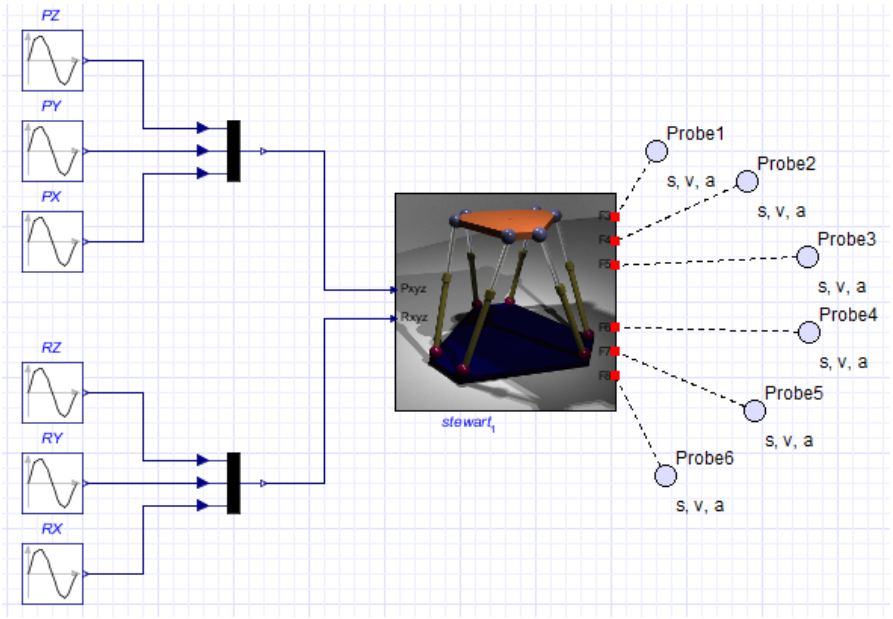


图 12-71　Stewart 子程序图标

12.2.6 查看仿真数据

运行模型，运行结束将弹出仿真结果。

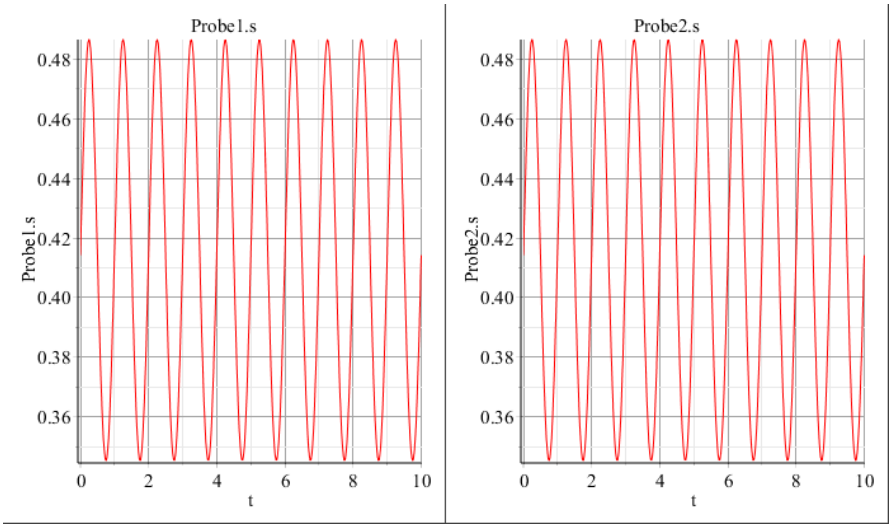


图 12-72　仿真结果

12.3 练习 3：Stewart 平台动力学仿真

概要

动力学是研究物体的运动和作用力之间的关系。Stewart 动力学仿真为获得六个分支相应运动下的所受到的作用力关系。相比于运动学仿真，动力学仿真的主动件为六个连杆，从动件为上平台。

为获模型特定运动下的动力学仿真结果，传统的做法是通过运动学仿真输出连杆的位置数据，通过 spline 指令将运动学仿真的数据，作为连杆驱动的运动规律源，使得在连杆驱动时，上平台按期望轨迹运行，再测量出该运动下的连杆受力，该方法只能适合一个运动下的动力学仿真，改变运动规律则需要重复上述操作（如 Adams 动力学仿真）；

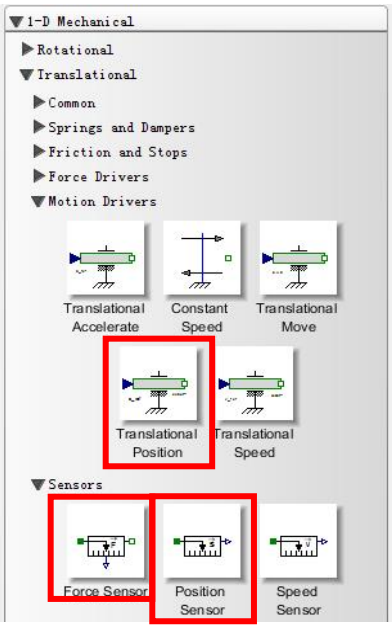
MapleSim 的动力学仿真，将运动学模型（获取分支伸长量数据）和动力学模型（分支为主动件驱动模型）统一即双 Stewart 模型，可以实现不同运动下的动力学仿真。

主要内容有：

1.指定连杆驱动：利用 Stewart 运动学仿真模型获得杆长的驱动数据，通过 Translation position 驱动连杆运动，驱动 Stewart 动力学模型。

2.测量连杆受力：通过 Force Sensor 测量连杆运动时的受力；

该实例将用到的元件：



Translational Position:用于指定移动副的位置

Force Sensor：用于测量移动副的受力

Position Sensor:用于测量移动副的位置

图 12-73　实例中的元件

12.3.1 创建杆长驱动模型

1.将 stewartkin 另存为 stewartdyn(dynamic)用于动力学仿真。

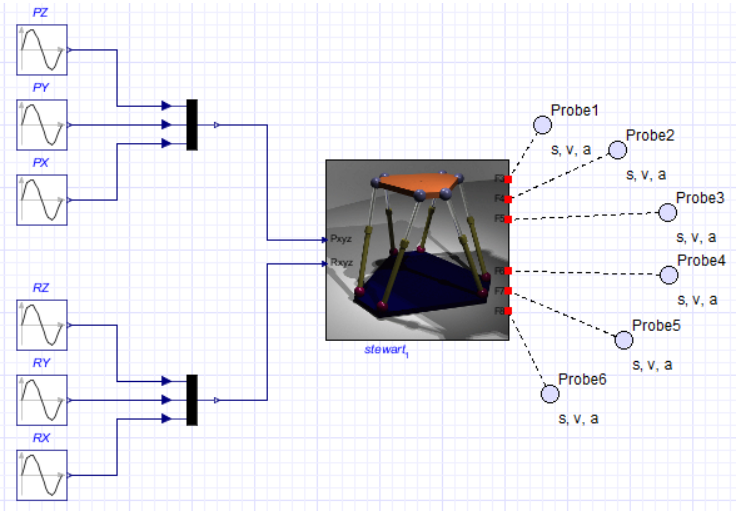


图 12-74　stewartdyn 模型

2.将所有的探针删除，接着复制 stewart 子系统，选择为独立子系统

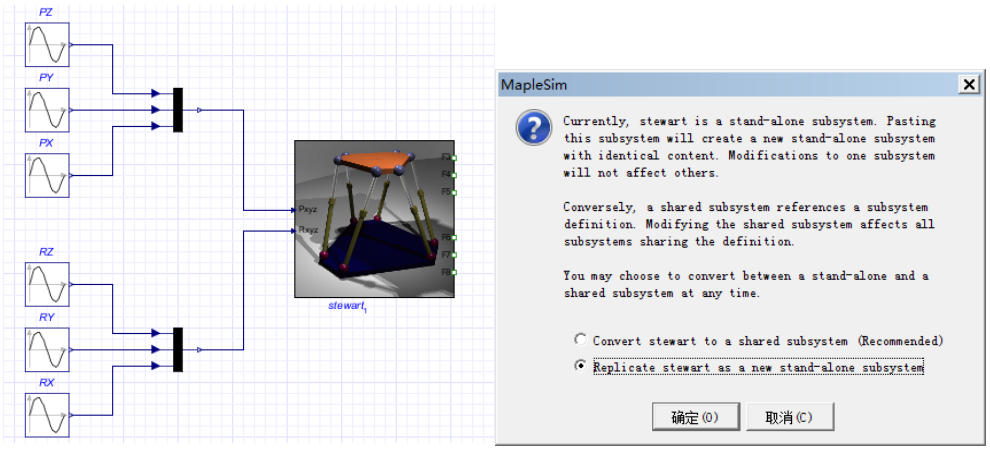


图 12-75　复制 stewart 子系统为独立子系统

3.复制后的系统如下，进入第二个 Stewart 平台的上平台子系统，将上平台的驱动删除。第一个 Stewart 模型为上平台驱动，第二个 Stewart 模型为杆长驱动。

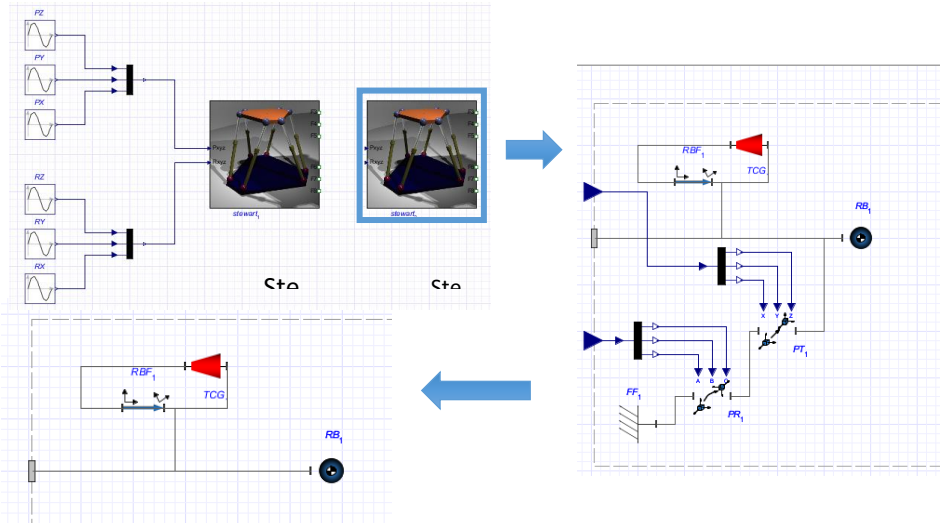


图 12-76　删除上平台驱动元件

4.退回到 sewart$_2$ 子系统目录，将六个分支和连线以及端口删除，只保留上平台模型，如下图所示。

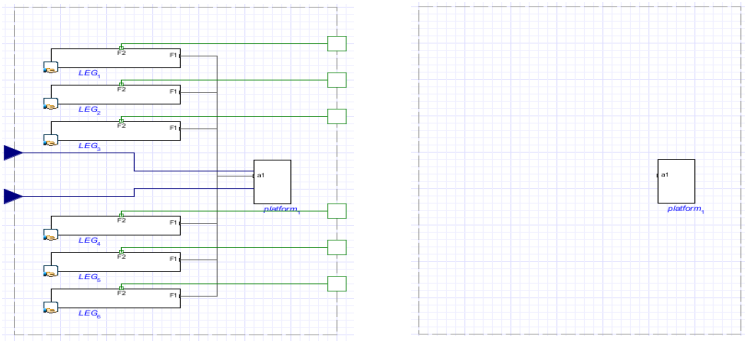


图 12-77　删除分支以及连线

5.打开 Local Component 目录，将 Component 目录下的共享子系统 LEG 复制，操作方法：选中 LEG 右击鼠标旋转 Duplicate，在弹出的命名框里输入 LEGtwo。该子系统用于创建第二个 Stewart 机构的六个分支，该操作使得 LEG 和 LEGtwo 相互独立。

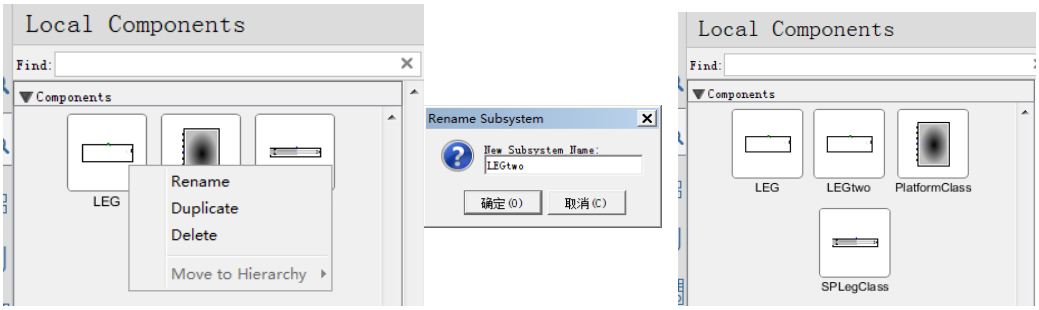


图 12-78　复制分支 LEGtwo

6.拉出 6 个 LEGtwo 子系统到 stewart$_2$ 子系统中机构的。并将其参数修改如下：如 LEGtwo1 参数设为 XA=XA1,YA=YA1,ZA=ZA1 等。

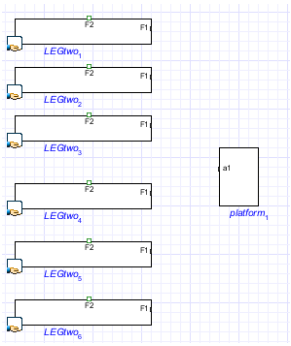


图 12-79　复制到六个 LEGtwo 子系统

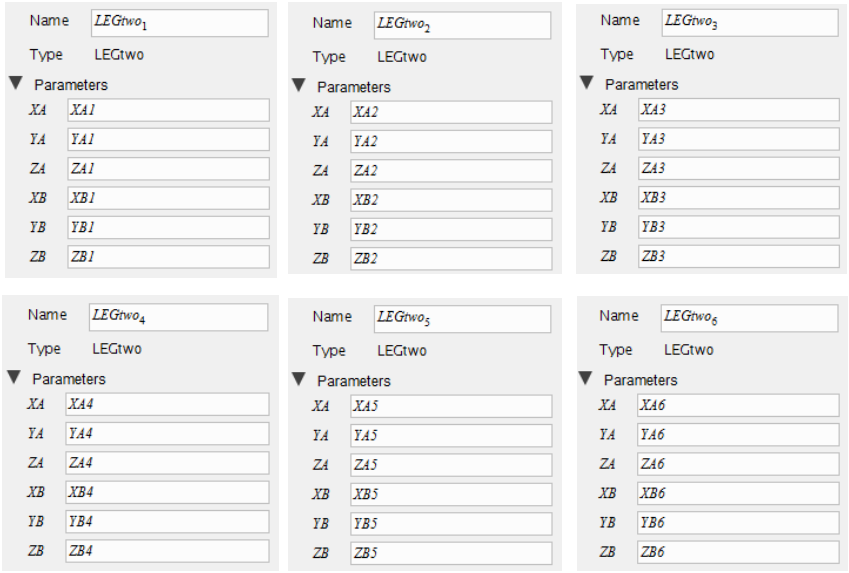


图 12-80　修改 LEGtwo 的参数

7.将 6 个分支与上平台连接如下图所示，进入 LEGtwo 子系统，将移动副到端口上的连线删除。

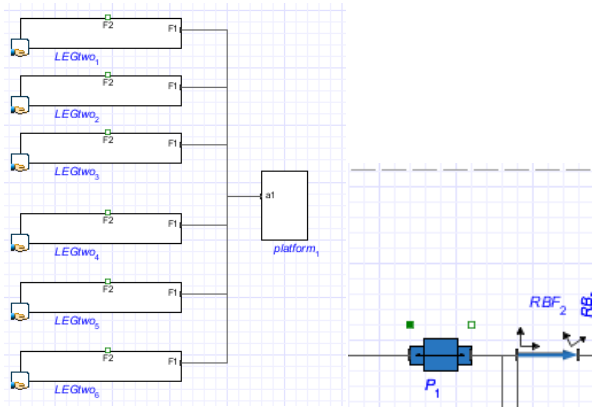


图 12-81　删除 P 副输出连线

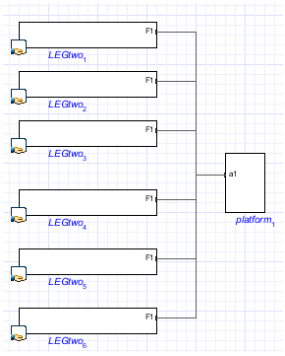


图 12-82　连接分支与上平台

8.拉出 Translational Position 和 Force Sensor 到 LEGtwo 子系统中，勾选 Translational Position 参数的 exact，指定输入数据为精确输入，并按下图连线。该操作指定了第二个 stewart 模型的输入为杆长驱动，并把所输出的力测量出来。

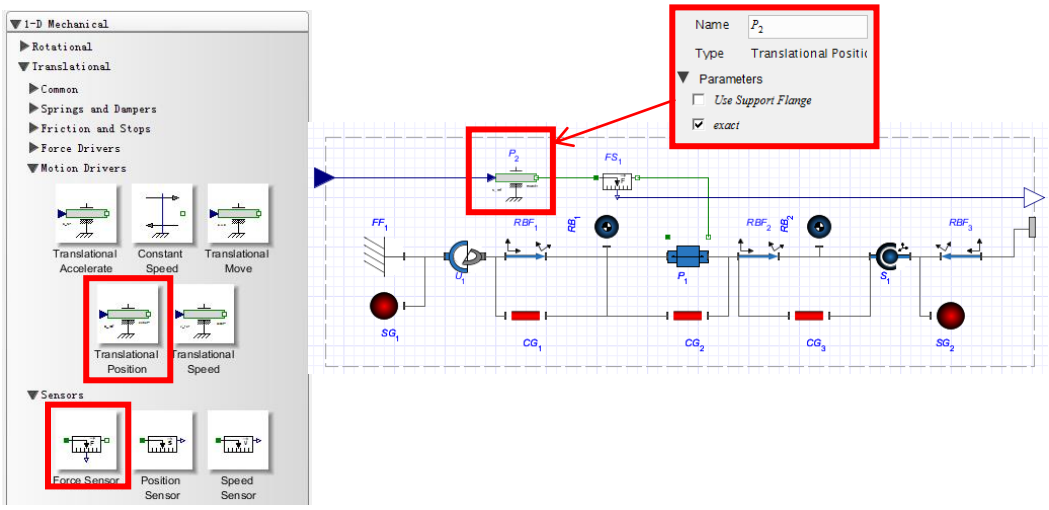


图 12-83　指定分支为理想位置驱动

9.将杆长输入端口连接到 $stewart_2$ 子系统的左端，将力矩输出端口连接到 $stewart_2$ 子系统的右端，进入上平台子系统 platform，将上平台质心的类型设置为 Treat as guess，上平台的位置由杆长决定，如下图所示。

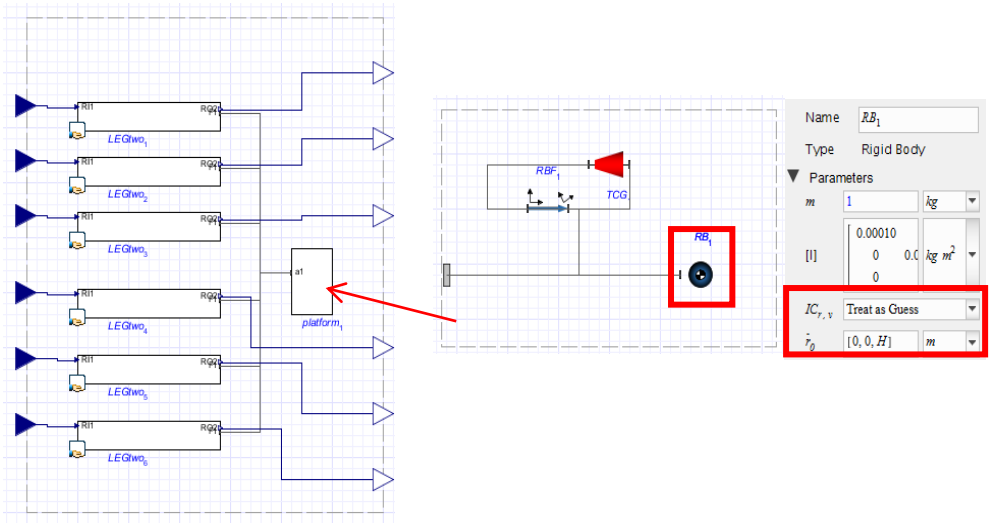
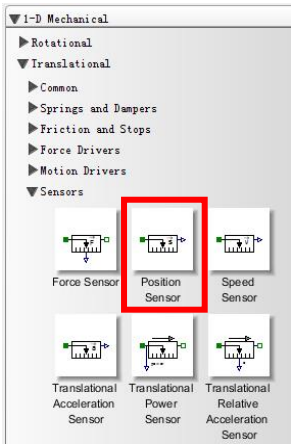


图 12-84　修改上平台的初始值类型

12.3.2 模型动力学仿真

1.在主目录下，添加 6 个 Position Sensor 用于测量第一个 Stewart 平台运动时各分支连杆的位置量，并将该位置值作为第二个 Stewart 平台的杆长输入值。



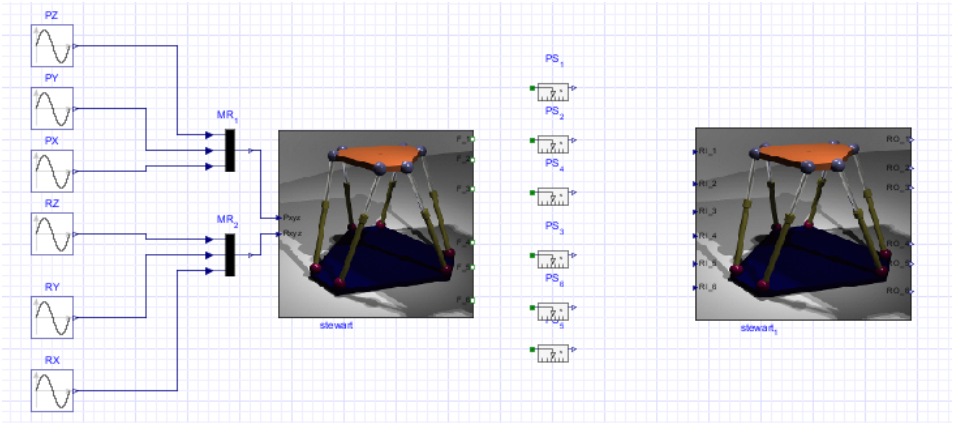


图 12-85　添加位置测量传感器元件

2.依序连接第一个 Stewart 模型输出和第二个 Stewart 模型输入。

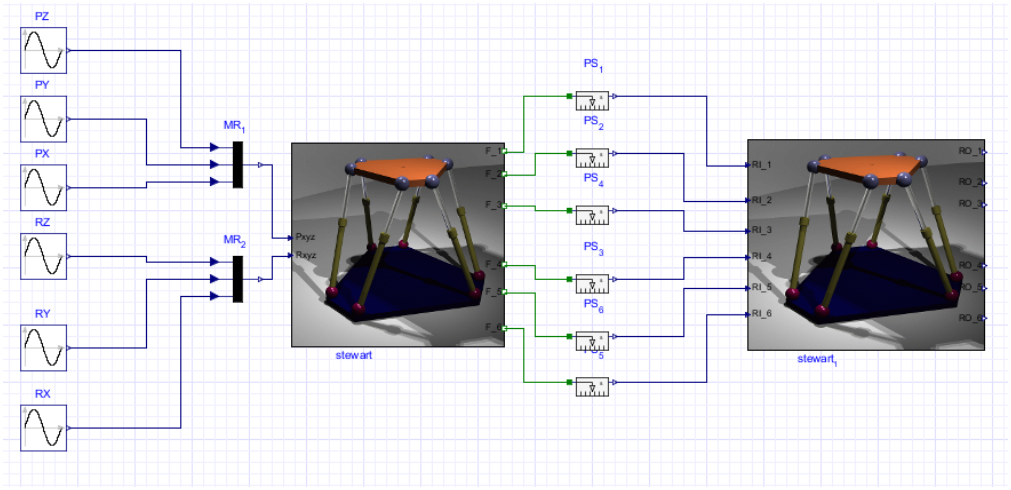


图 12-86　连接两个 Stewart 模型

4.使用探针 probe 测量六个分支的受力，将探针命名为 force1~force6。

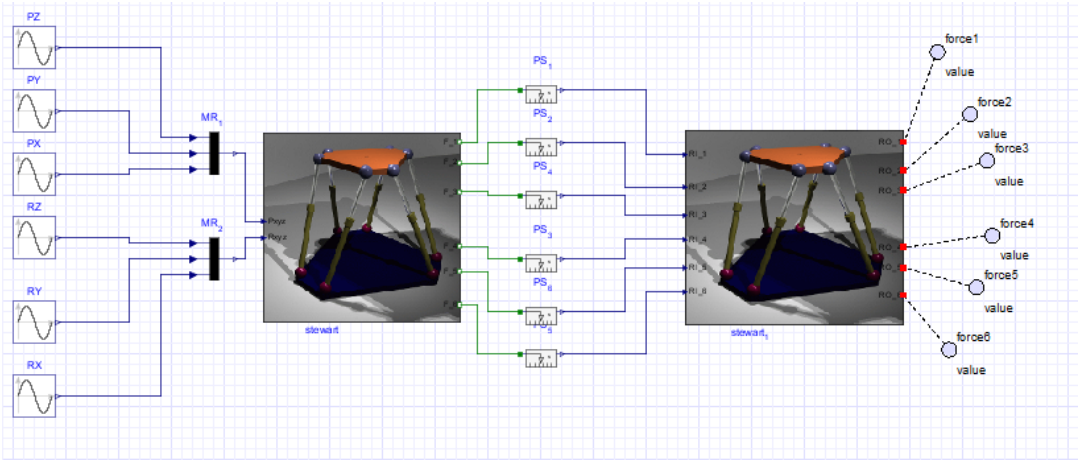


图 12-87　测量连杆受力

12.3.3 查看仿真结果

运行模型，可查看六个分支的驱动力，根据输出的结果可用于选择驱动电机的输出，验证动力学推导的正确性，以及动力学控制等。

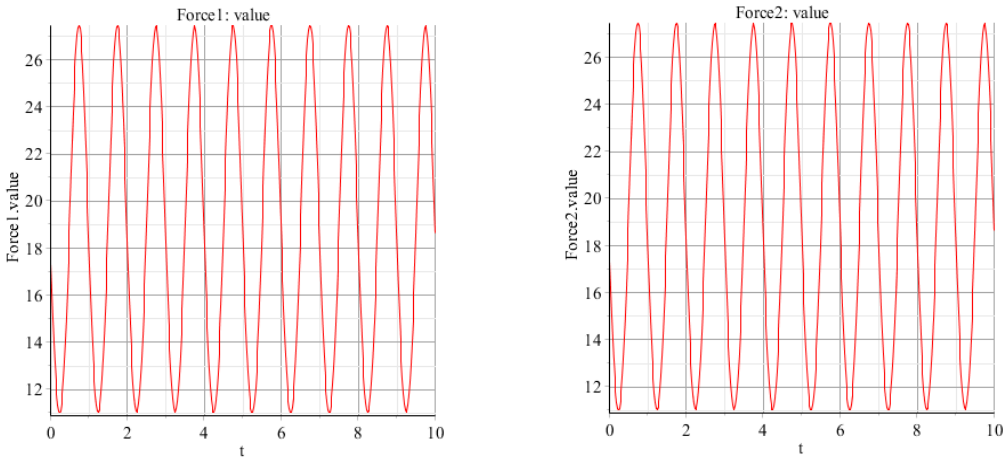


图 12-88　stewartdyn 模型动力学仿真结果

12.4 练习 4：Stewart 平台运动控制

概要

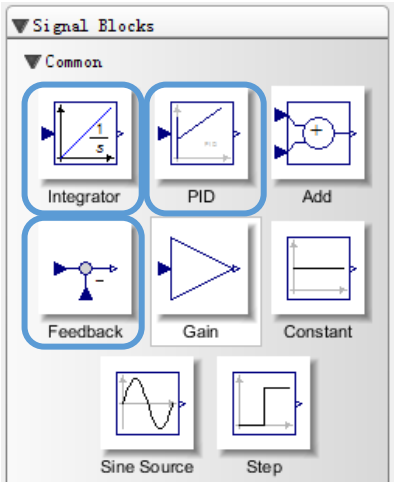
运动控制是使机械运动实现精确的位置控制、速度控制、加速度控制、力矩或力的控制、以及这些被控量的综合控制。

Stewart 平台运动控制，为控制六个主动分支按特定规律运行，从而实现上平台沿着期望的轨迹运动。控制上有基于位置、速度、力矩控制三种方式，本节模型采用位置控制和力矩控制两种方式。

1.模型逆运动学方程：在 Stewart 运动控制中，为实现平台按期望轨迹运行，需要求出该该轨迹下各杆长的位置量，为逆运动学求解的过程。

2.上平台输出测量：对模型以期望轨迹运行后，为了验证控制效果，需要测量上平台的位置、姿态输出结果，并与期望输入进行对比，进而优化控制策略。

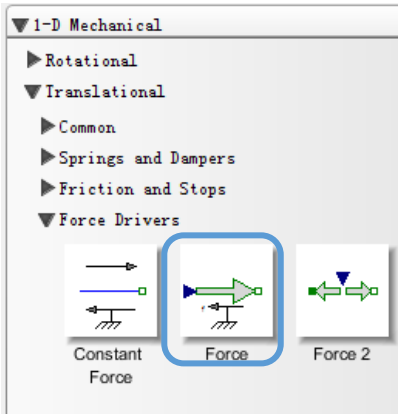
该实例将用到的元件：



Integrator:积分元件

PID:PID 调节元件

Feedback:反馈输入元件



Force:力驱动元

图 12-89　实例中的元件

12.4.1 模型多体分析

1. 打开练习 1 中的 stewartmodel 模型，另存为 stewartcontrol ，点击左侧项目面板中的“Add Apps and Templates”（  ）,在 Templates 中选择 Worksheet，为模型添加分析模板,双击后点击（  ）进入 Maple。

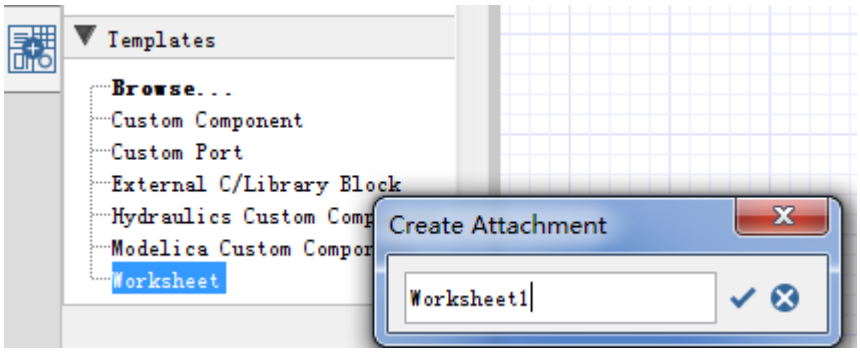


图 12-90　添加分析模板

2. 输入以下程序：

```
mModel := MapleSim:-LinkModel();

MB := mModel:-GetMultibody(simplify = true);
```

运行该模板，可以获得 stewart 模型的运动学方程和动力学方程。

注意：换行为 Shift+Enter，运行为 Enter 或点击（ ）。

3. 运行结束可以看到 stewart 模型的自由度为 6，采用 24 个变量 18 个约束方程描述模型运动学方程和动力学方程。

```
"Analyzing system..."
"Performing constraint analysis..."
"The system has 6 degree(s) of freedom.  It is modeled using 24 generalized coordinate(s) coupled by 18 algebraic constraint(s)."
"Peforming a dynamic analysis using an augmented reaction formulation - system variables shown below:"
"vQ", [1 .. 24 Vector_column, Data Type: anything, Storage: rectangular, Order: Fortran_order], "vP", [1 .. 24 Vector_column, Data Type: anything, Storage: rectangular, Order: Fortran_order], "vPdot", [1 .. 24 Vector_column, Data Type: anything, Storage: rectangular, Order: Fortran_order], "vReactions", [1 .. 18 Vector_column, Data Type: anything, Storage: rectangular, Order: Fortran_order]
"Dynamic analysis complete."
```

图 12-91　Stewart 模型运动学和动力学约束方程

4. 在 Multibody Analysis 输入以下程序：

```
vPosCons:=MB:-GetPosCons();
```

按 Enter 键可将位置约束方程显示。

```
> vPosCons := MB:-GetPosCons( );

vPosCons := [1 .. 18 Vector_column, Data Type: anything, Storage: rectangular, Order: Fortran_order]
```

图 12-92　显示约束方程

5. 将鼠标放在 vPosCons:=MB:-GetPosCons();上，点击上面的插入命令行指令 ，插入三个指令输入行，如下图所示。

```
> vPosCons := MB:-GetPosCons( );

                                  vPosCons := [ 1 .. 18 Vector_column
                                                Data Type: anything
                                                Storage: rectangular
                                                Order: Fortran_order ]
>
>
>
```

图 12-93　插入命令行

6. 在指令行上输入

```
eq1:=vPosCons(1)=0;

eq2:=vPosCons(2)=0;

eq3:=vPosCons(3)=0;
```

然后依次按 Enter 执行这三个命令，用于获取模型位置约束方程的前三个方程,这三个方程描述了分支一的位置约束方程。由于 Stewart 机构的分支结构是一样，因此该方程组可以通过修改上下平台坐标值参数用于描述其他分支的位置约束方程。

7. 分支一的位置约束方程如下。含有上平台移动变量 x(t),y(t),z(t),转动变量 $\zeta(t),\eta(t),\xi(t)$。分支一的 U 副两个转角变量 LEG1_U1_alpha(t),LEG1_U1_beta(t)和移动副 P 变量 LEG1_P1_s(t)。

```
> eq1:=vPosCons(1)=0;
eq1 := -sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(η(t)) sin(ξ(t)) sin(ζ(t)) YA1 + sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(η(t)) cos(ξ(t)) sin(ζ(t)) XA1
    - sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) sin(η(t)) sin(ξ(t)) cos(ζ(t)) YA1 + sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) sin(η(t)) cos(ξ(t)) cos(ζ(t)) XA1
    + sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(ξ(t)) cos(ζ(t)) XA1 + sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(ξ(t)) cos(ζ(t)) YA1
    - sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(η(t)) sin(ζ(t)) ZA1 - sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) sin(ξ(t)) sin(ζ(t)) XA1
    - sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) cos(ξ(t)) sin(ζ(t)) YA1 - sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) cos(η(t)) cos(ζ(t)) ZA1
    - cos(LEG1_U1_beta(t)) sin(ξ(t)) cos(η(t)) YA1 + cos(LEG1_U1_beta(t)) cos(ξ(t)) cos(η(t)) XA1 + y(t) sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t))
    - z(t) sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) - YB1 sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) + ZB1 sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t))
    + cos(LEG1_U1_beta(t)) sin(η(t)) ZA1 + x(t) cos(LEG1_U1_beta(t)) - XB1 cos(LEG1_U1_beta(t)) = 0
> eq2:=vPosCons(2)=0;
eq2 := sin(LEG1_U1_alpha(t)) sin(η(t)) sin(ξ(t)) cos(ζ(t)) YA1 - sin(LEG1_U1_alpha(t)) sin(η(t)) cos(ξ(t)) cos(ζ(t)) XA1 - cos(LEG1_U1_alpha(t)) sin(η(t)) sin(ξ(t)) sin(ζ(t)) YA1
    + cos(LEG1_U1_alpha(t)) sin(η(t)) cos(ξ(t)) sin(ζ(t)) XA1 + sin(LEG1_U1_alpha(t)) sin(ξ(t)) sin(ζ(t)) XA1 + sin(LEG1_U1_alpha(t)) cos(ξ(t)) sin(ζ(t)) YA1
    + sin(LEG1_U1_alpha(t)) cos(η(t)) cos(ζ(t)) ZA1 + cos(LEG1_U1_alpha(t)) sin(ξ(t)) cos(ζ(t)) XA1 + cos(LEG1_U1_alpha(t)) cos(ξ(t)) cos(ζ(t)) YA1
    - cos(LEG1_U1_alpha(t)) cos(η(t)) sin(ζ(t)) ZA1 + y(t) cos(LEG1_U1_alpha(t)) + z(t) sin(LEG1_U1_alpha(t)) - ZB1 sin(LEG1_U1_alpha(t)) - YB1 cos(LEG1_U1_alpha(t)) = 0
> eq3:=vPosCons(3)=0;
eq3 := cos(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) sin(η(t)) sin(ξ(t)) cos(ζ(t)) YA1 + cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(η(t)) sin(ξ(t)) sin(ζ(t)) YA1
    - cos(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) sin(η(t)) cos(ξ(t)) cos(ζ(t)) XA1 - cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(η(t)) cos(ξ(t)) sin(ζ(t)) XA1
    + cos(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) sin(ξ(t)) sin(ζ(t)) XA1 - cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(ξ(t)) cos(ζ(t)) XA1
    + cos(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) cos(ξ(t)) sin(ζ(t)) YA1 - cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(ξ(t)) cos(ζ(t)) YA1
    + cos(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) cos(η(t)) cos(ζ(t)) ZA1 + cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(η(t)) sin(ζ(t)) ZA1
    - sin(LEG1_U1_beta(t)) sin(ξ(t)) cos(η(t)) YA1 + sin(LEG1_U1_beta(t)) cos(ξ(t)) cos(η(t)) XA1 - y(t) cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t))
    + z(t) cos(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) + sin(LEG1_U1_beta(t)) sin(η(t)) ZA1 - ZB1 cos(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t))
    + YB1 cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) + x(t) sin(LEG1_U1_beta(t)) - XB1 sin(LEG1_U1_beta(t)) - LEG1_P1_s(t) - 1 = 0
```

图 12-94　Stewart 模型位置约束方程提取

12.4.2 自定义逆运动学方程

1. 在下 eq1:=vPosCons(1)=0;面再插入三个命令行并按如下输入:

eq1:=subs(zeta(t)=A(t),eta(t)=B(t),xi(t)=C(t),eq1);

eq2:=subs(zeta(t)=A(t),eta(t)=B(t),xi(t)=C(t),eq2);

eq3:=subs(zeta(t)=A(t),eta(t)=B(t),xi(t)=C(t),eq3);

2. Subs()命令用于替换变量名称，zeta(t)-A(t),eta(t)=B(t),xi(t)=C(t) 目的是将$\zeta(t)$替换为 A(t)，$\eta(t)$替换为 B(t)，$\xi(t)$替换为 C(t)。A,B,C 分别为上平台绕 X 轴，Y 轴和 Z 轴的转角，使用该替换为了便于识别。

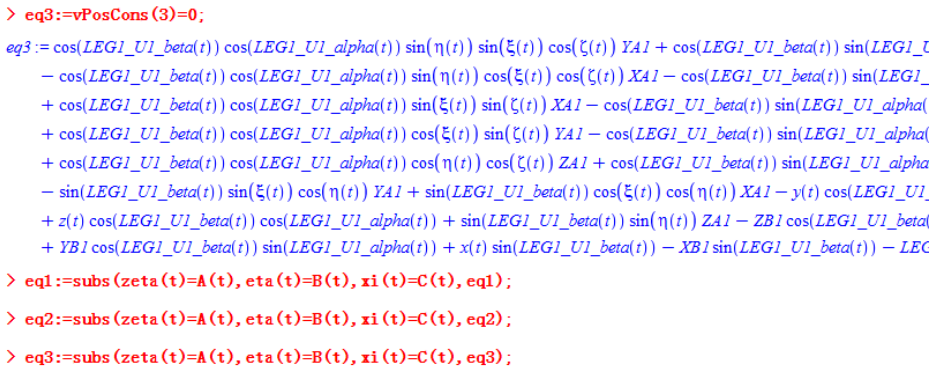


图 12-95　Stewart 模型位置约束方程变量替换

3. 再插入一个命令行并输入:

eq:=[eq1,eq2,eq3];

将三个方程组成一个数组，执行这四个命令行。

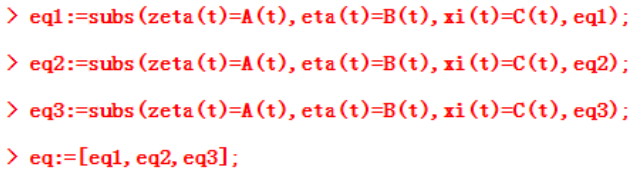


图 12-96　输入变换替换指令

4. 执行后的方程如下,保存执行后多体分析模板。

```
> eq:=[eq1, eq2, eq3];
eq := [ -sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(B(t)) sin(C(t)) sin(A(t)) I
    - sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) sin(B(t)) sin(C(t)) cos(A(t)) I
    + sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(C(t)) cos(A(t)) XA1 + sin(
    - sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(B(t)) sin(A(t)) ZA1 - sin(.
    - sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) cos(C(t)) sin(A(t)) YA1 - sin(
    - cos(LEG1_U1_beta(t)) sin(C(t)) cos(B(t)) YA1 + cos(LEG1_U1_beta(t)) cos(
    - z(t) sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) - YB1 sin(LEG1_U1_beta
    + cos(LEG1_U1_beta(t)) sin(B(t)) ZA1 + x(t) cos(LEG1_U1_beta(t)) - XB1 cos
    - sin(LEG1_U1_alpha(t)) sin(B(t)) cos(C(t)) cos(A(t)) XA1 - cos(LEG1_U1_al
    + sin(LEG1_U1_alpha(t)) sin(C(t)) sin(A(t)) XA1 + sin(LEG1_U1_alpha(t)) cos
    + cos(LEG1_U1_alpha(t)) sin(C(t)) cos(A(t)) XA1 + cos(LEG1_U1_alpha(t)) cc
    + z(t) sin(LEG1_U1_alpha(t)) - ZB1 sin(LEG1_U1_alpha(t)) - YB1 cos(LEG1_
    + cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(B(t)) sin(C(t)) sin(A(t)) Y
    - cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(B(t)) cos(C(t)) sin(A(t)) X
    - cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(C(t)) cos(A(t)) XA1 + cos
    - cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(C(t)) cos(A(t)) YA1 + cos
    + cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(B(t)) sin(A(t)) ZA1 - sin(
    - y(t) cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) + z(t) cos(LEG1_U1_beta
    - ZB1 cos(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) + YB1 cos(LEG1_U1_bet
    = 0]
```

图 12-97　变换替换后方程组

5. 按 Ctrl+C 复制替换后组成的三个方程 eq:=[eq1,eq2,eq3];，作为分支的逆运动学方程。

```
> eq:=[eq1, eq2, eq3];
eq := [ -sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(B(t)) sin(C(t)) sin(A(t)) YA1 + sin(LEG1_U1
    - sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) sin(B(t)) sin(C(t)) cos(A(t)) YA1 + sin(LEG1_U1
    + sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(C(t)) cos(A(t)) XA1 + sin(LEG1_U1_beta(t)) s
    - sin(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(B(t)) sin(A(t)) ZA1 - sin(LEG1_U1_beta(t)) c
    - sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) cos(C(t)) sin(A(t)) YA1 - sin(LEG1_U1_beta(t)) c
    - cos(LEG1_U1_beta(t)) sin(C(t)) cos(B(t)) YA1 + cos(LEG1_U1_beta(t)) cos(C(t)) cos(B(t)) XA1
    - z(t) sin(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) - YB1 sin(LEG1_U1_beta(t)) sin(LEG1_U1_al
    + cos(LEG1_U1_beta(t)) sin(B(t)) ZA1 + x(t) cos(LEG1_U1_beta(t)) - XB1 cos(LEG1_U1_beta(t))
    - sin(LEG1_U1_alpha(t)) sin(B(t)) cos(C(t)) cos(A(t)) XA1 - cos(LEG1_U1_alpha(t)) sin(B(t)) sin
    + sin(LEG1_U1_alpha(t)) sin(C(t)) sin(A(t)) XA1 + sin(LEG1_U1_alpha(t)) cos(C(t)) sin(A(t)) YA
    + cos(LEG1_U1_alpha(t)) sin(C(t)) cos(A(t)) XA1 + cos(LEG1_U1_alpha(t)) cos(C(t)) cos(A(t)) YA
    + z(t) sin(LEG1_U1_alpha(t)) - ZB1 sin(LEG1_U1_alpha(t)) - YB1 cos(LEG1_U1_alpha(t)) = 0, co
    + cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(B(t)) sin(C(t)) sin(A(t)) YA1 - cos(LEG1_U1_
    - cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(B(t)) cos(C(t)) sin(A(t)) XA1 + cos(LEG1_U1_
    - cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) sin(C(t)) cos(A(t)) XA1 + cos(LEG1_U1_beta(t)) c
    - cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(C(t)) cos(A(t)) YA1 + cos(LEG1_U1_beta(t)) c
    + cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) cos(B(t)) sin(A(t)) ZA1 - sin(LEG1_U1_beta(t)) s
    - y(t) cos(LEG1_U1_beta(t)) sin(LEG1_U1_alpha(t)) + z(t) cos(LEG1_U1_beta(t)) cos(LEG1_U1_a
    - ZB1 cos(LEG1_U1_beta(t)) cos(LEG1_U1_alpha(t)) + YB1 cos(LEG1_U1_beta(t)) sin(LEG1_U1_a
    = 0]
```

图 12-98　复制分支一方程组

6. 点击  ,从中选择 Custom Component(用户自定义元件)，为模型添加自定义元

件,点击创建模板 Create Attachment 进入到 maple 环境。

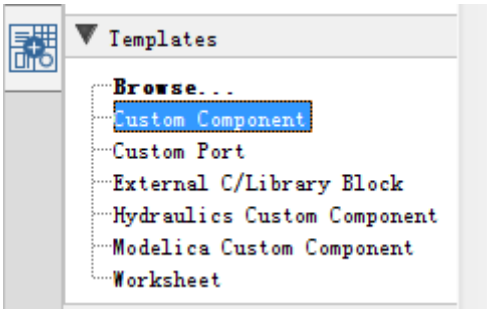


图 12-99　添加自定义元件分析模板

7. 在输入方程处，删除掉[];并将已复制的方程组粘贴到此，按 Enter 键执行该方程组。

Define Equations:

$$
\begin{aligned}
eq := [&-\sin(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))\sin(B(t))\sin(C(t))\sin(A(t))\,YA1\\
&-\sin(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t))\sin(B(t))\sin(C(t))\cos(A(t))\,YA1+\\
&+\sin(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))\sin(C(t))\cos(A(t))\,XA1+\sin(LEG\\
&-\sin(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))\cos(B(t))\sin(A(t))\,ZA1-\sin(LEG\\
&-\sin(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t))\cos(C(t))\sin(A(t))\,YA1-\sin(LEG\\
&+\cos(LEG1\_U1\_beta(t))\cos(C(t))\cos(B(t))\,XA1+y(t)\sin(LEG1\_U1\_beta(t))\sin(L\\
&-YB1\sin(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))+ZB1\sin(LEG1\_U1\_beta(t))\,c\\
&-XB1\cos(LEG1\_U1\_beta(t))=0,\sin(LEG1\_U1\_alpha(t))\sin(B(t))\sin(C(t))\cos(A(t\\
&-\cos(LEG1\_U1\_alpha(t))\sin(B(t))\sin(C(t))\sin(A(t))\,YA1+\cos(LEG1\_U1\_alpha(t\\
&+\sin(LEG1\_U1\_alpha(t))\cos(C(t))\sin(A(t))\,YA1+\sin(LEG1\_U1\_alpha(t))\cos(B(t\\
&+\cos(LEG1\_U1\_alpha(t))\cos(C(t))\cos(A(t))\,YA1-\cos(LEG1\_U1\_alpha(t))\cos(B(\\
&-YB1\cos(LEG1\_U1\_alpha(t))=0,\cos(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t))\,si\\
&+\cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))\sin(B(t))\sin(C(t))\sin(A(t))\,YA1-\\
&-\cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))\sin(B(t))\cos(C(t))\sin(A(t))\,XA1+\\
&-\cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))\sin(C(t))\cos(A(t))\,XA1+\cos(LEG\\
&-\cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))\cos(C(t))\cos(A(t))\,YA1+\cos(LEC\\
&+\cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))\cos(B(t))\sin(A(t))\,ZA1-\sin(LEG\\
&-y(t)\cos(LEG1\_U1\_beta(t))\sin(LEG1\_U1\_alpha(t))+z(t)\cos(LEG1\_U1\_beta(t))\,c\\
&-ZB1\cos(LEG1\_U1\_beta(t))\cos(LEG1\_U1\_alpha(t))+YB1\cos(LEG1\_U1\_beta(t))
\end{aligned}
$$

图 12-100　复制分支方程到自定义元件方程栏

8. 点击 Parameters 下的 Refresh All 获取方程组的参数和变量，该方程组具有六个参数(上下平台的坐标)，9 个变量。

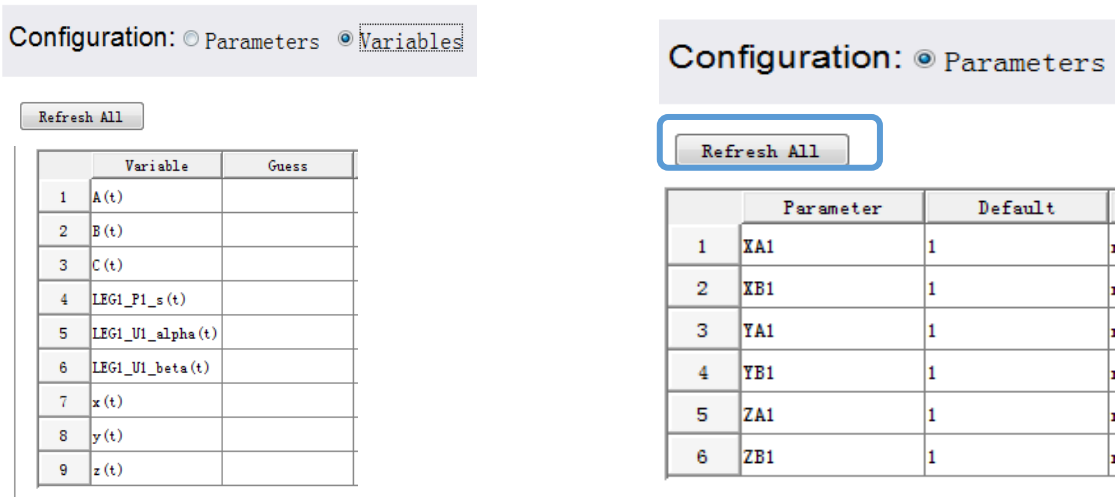


图 12-101　获取方程组变量和参数

9. 在 Ports 目录先清除所有端口，之后再添加三个端口。

图 12-102 添加端口

10. 调整端口位置，并修改第一个端口的属性，作为指定上平台 X,Y,Z 方向的移动量。

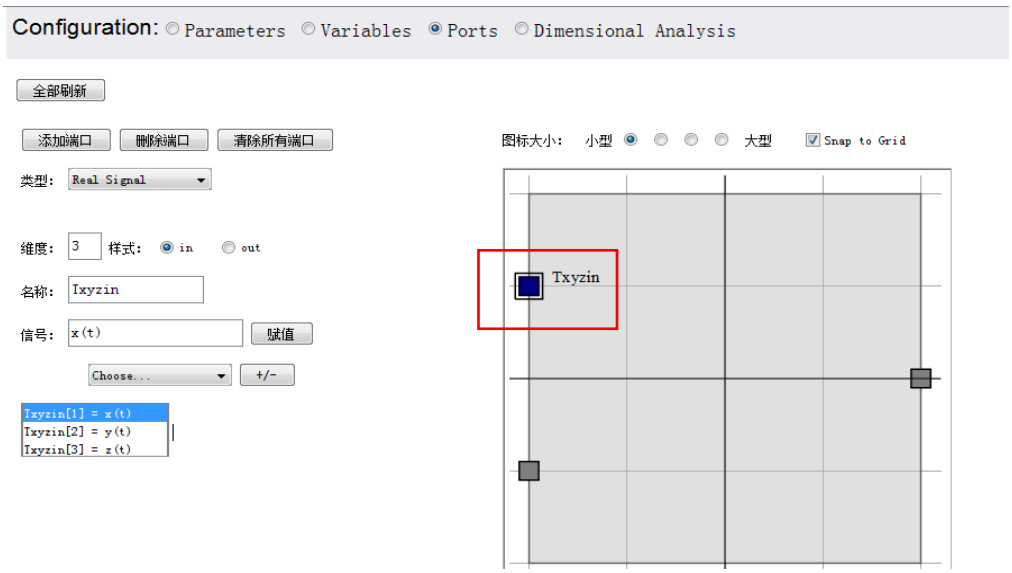


图 12-103　配置第一个端口 Txyzin

11. 配置第二个端口的属性，作为指定上平台 X,Y,Z 方向的转动量。

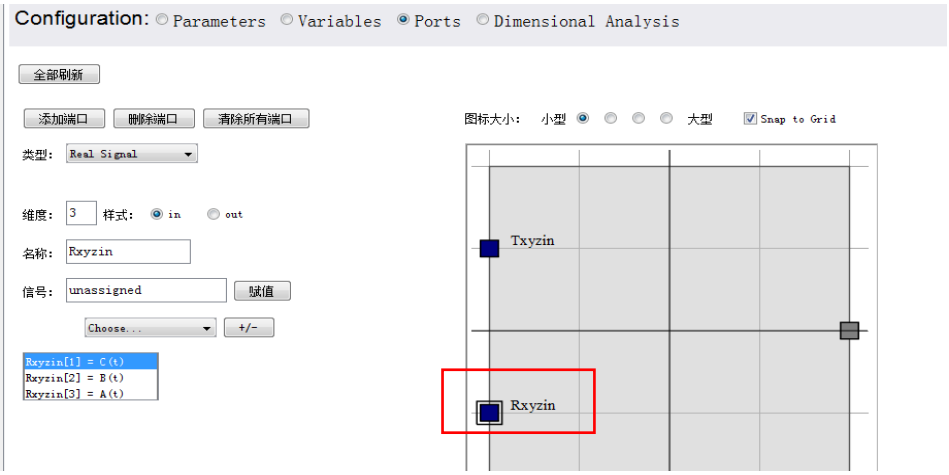


图 12-104　配置第二个端口 Rxyzin

12. 配置第三个端口的属性，作为对应上平台六个自由的运行下分支的伸长量。

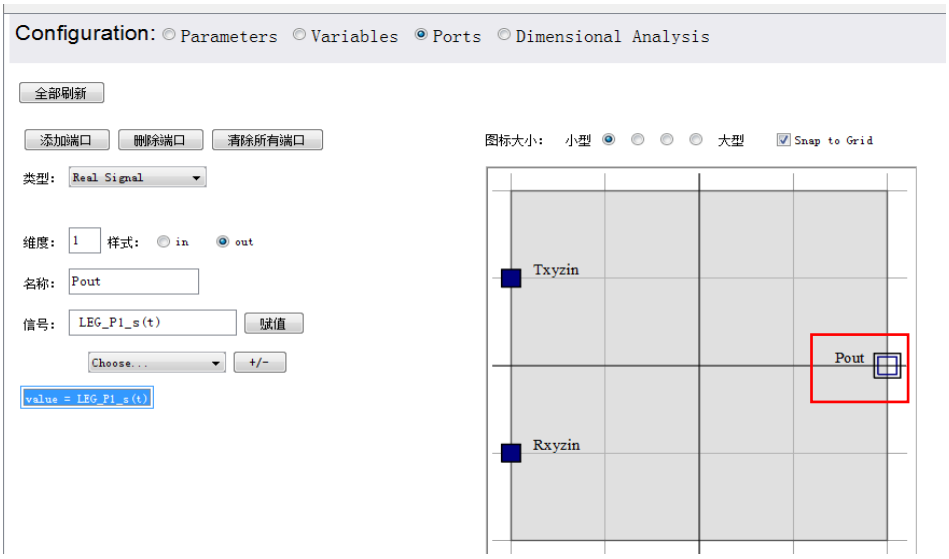


图 12-105　配置第三个端口 Pout

13. 将定义好的自定义元件名称修改为 invK(inverse kinematics 逆运动学的缩写)，并点击 Generate MapleSim Component 将该模式生成。

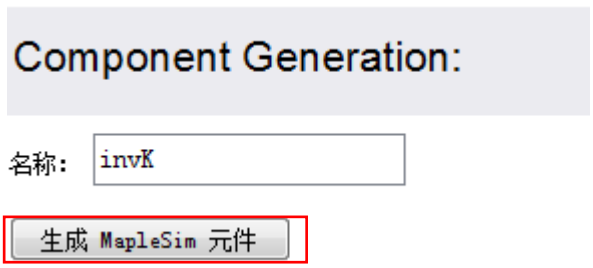


图 12-106　生成自定义元件 invK

14. 回到 maplesim 模型，在 Project->Definitions-> Components 目录下可以看到生成的 invK 的自定义元件。

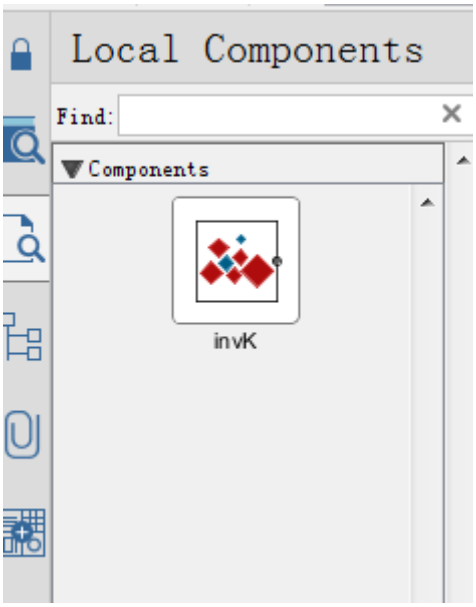


图 12-107　自定义元件 invK 生成

12.4.3 逆运动学控制

1. 进入 LEG 子系统，将刚生成逆运动学模型添加到子系统 LEG 中，修改 invK 模型的参数为：XA1=XA,XB1=XB,YA1=YA,YB1=YB,ZA1=ZA,ZB1=ZB，这样修改的参数可以接收分支的参数。

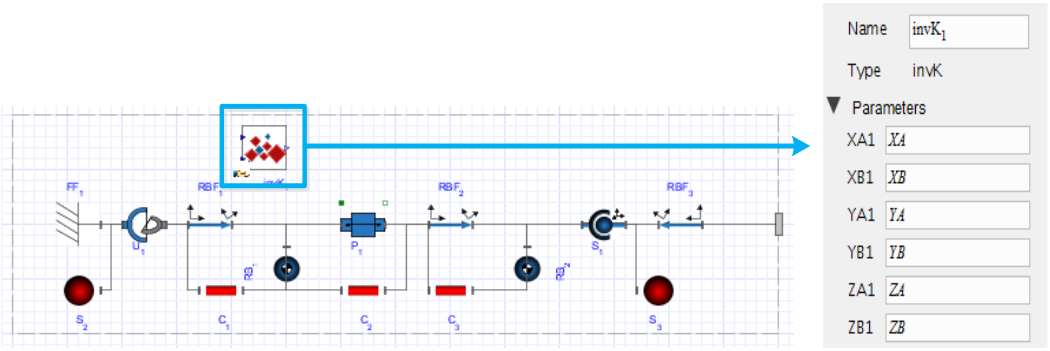


图 12-108　修改 invK 的参数

2. 添加 Translational Position 到该子系统中，并勾选 exact，指定输入为精确输入，并按如下图连接。

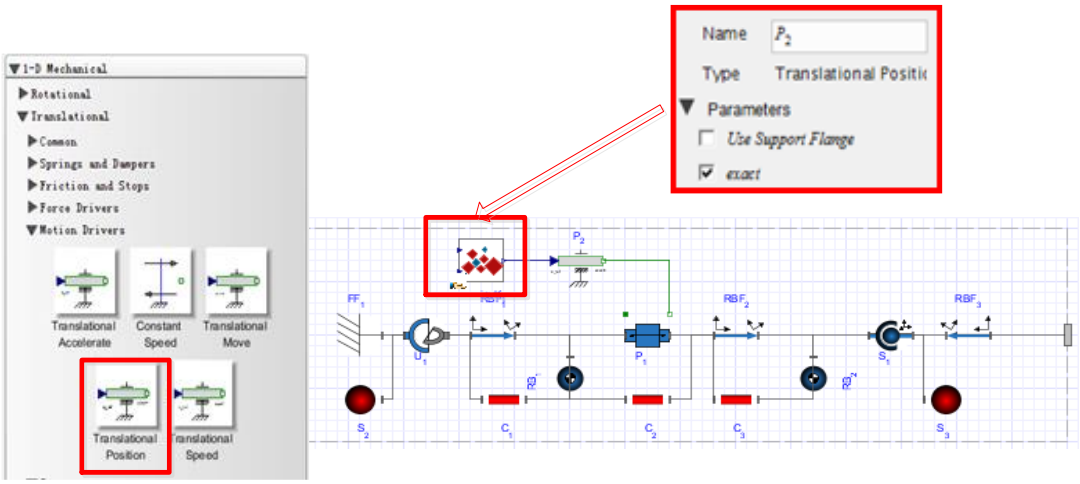


图 12-109　指定连杆精确位置驱动

3. 将自定义元件 invK 的输入端连接到该子系统的左端，如下图所示。

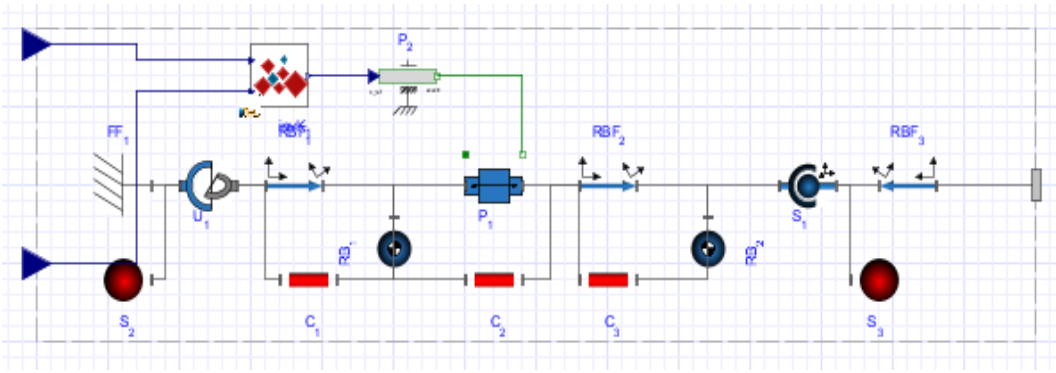


图 12-110　指定自定义元件输入端口

4. 退回到主系统，由于模型的位置由六个连杆长度决定，模型的初始位置应设定为 Treat as Guess 类型，如下图所示 。

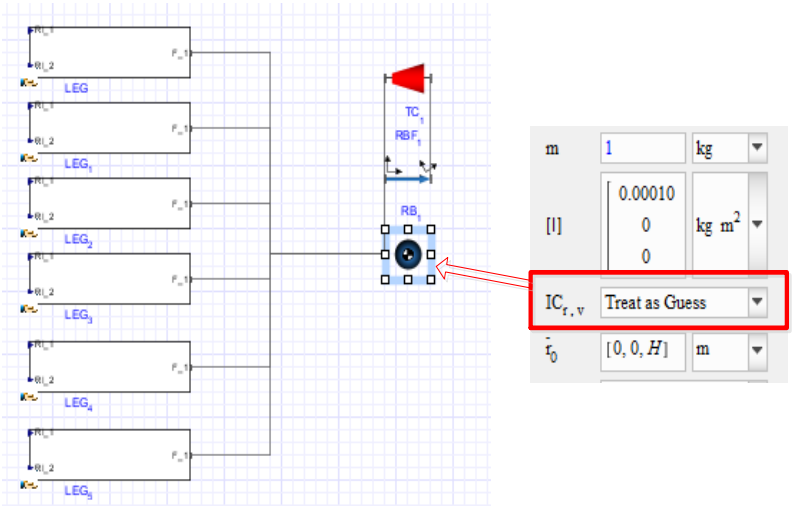


图 12-101　修改上平台初始位置类型

5. 全选 Stewart 连杆和上平台，创建 Stewart 子系统，如下图所示。

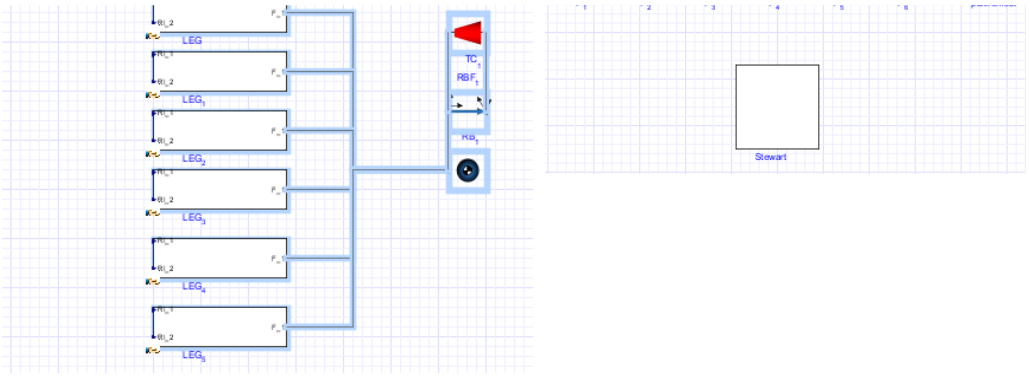


图 12-102　创建 Stewart 子系统

6. 在 Stewart 子系统下，将自定义元件的输入连接到子系统的左侧，如下图所示，将移动输入 R1-1 连接到同一个端口 RI1，转动输入 R1-2 连接到同一个端口 RI2,如下图所示。

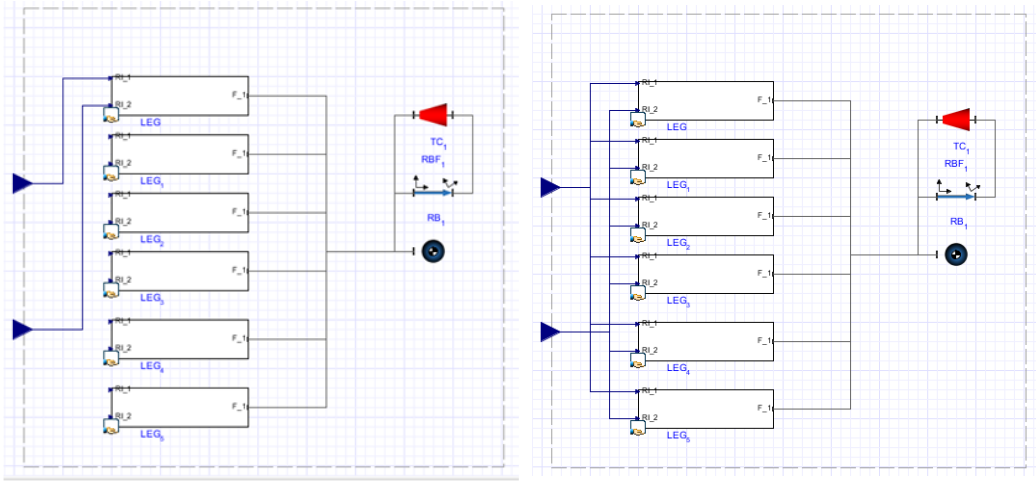


图 12-103　定义上平台的初始位置

7. 打开 Stewartkin 模型将输入复制到该模型中（也可以重建），将移动和转动输入对应连接，如下图所示；

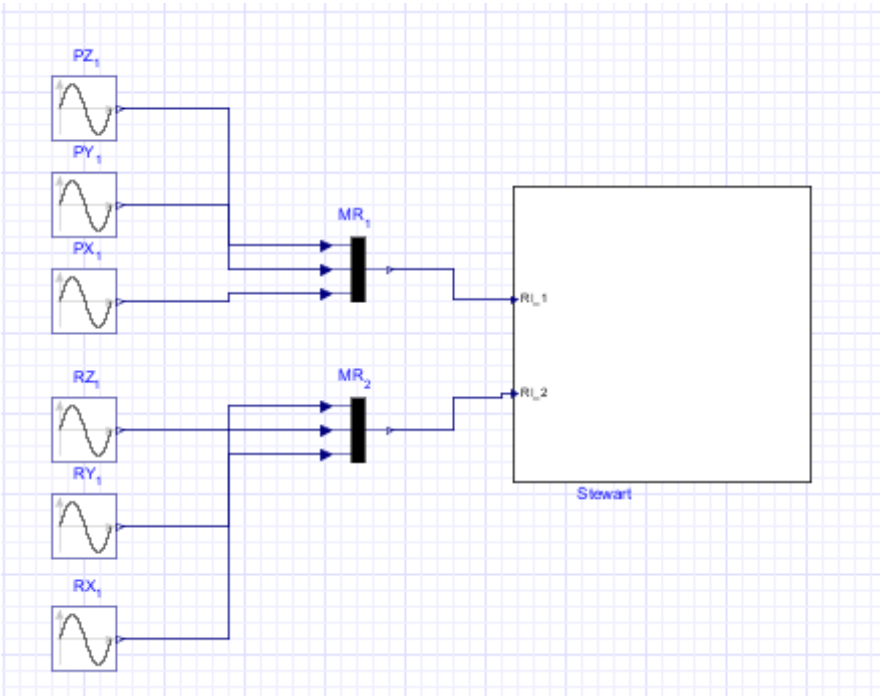


图 12-104　添加模型输入信号

8. 将 Z 轴移动的信号给定一个上平台的初始高度 H，即将 PZ 的信号给定一个偏置 offset=H ，使得给予的驱动是在初始位置(0,0,H)下的运动。

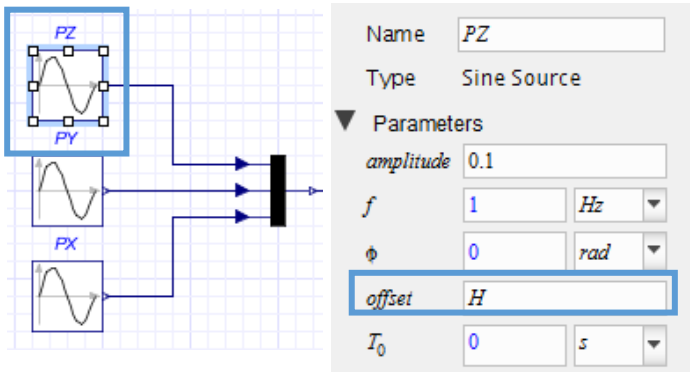


图 12-105 设定信号初始位置

9. 回到 stewartcontrol 模型，点击运行按钮  获得最新参数下的模型和运动效果。

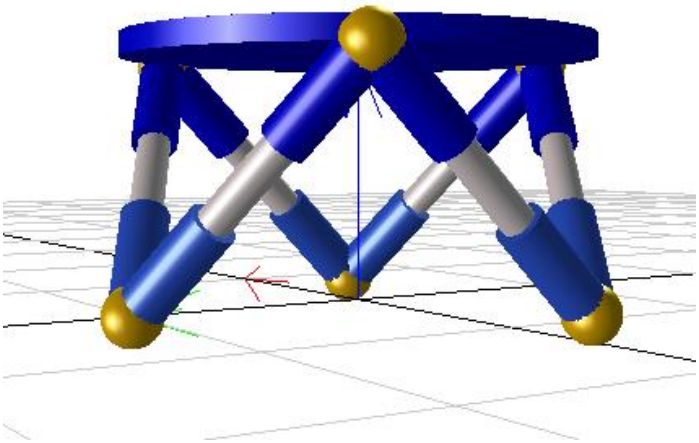


图 12-106　Stewart 跟踪平台

12.4.4 上平台位置姿态测量

1. 进入 Stewart 子系统，从在 Multibody->Sensors 目录下添加绝对位移传感器 Absolute Translation 和绝对转角传感器 Absolute Rotation，并分别连到上平台的质心端口上，用于测量上平台的位移量和转角值。

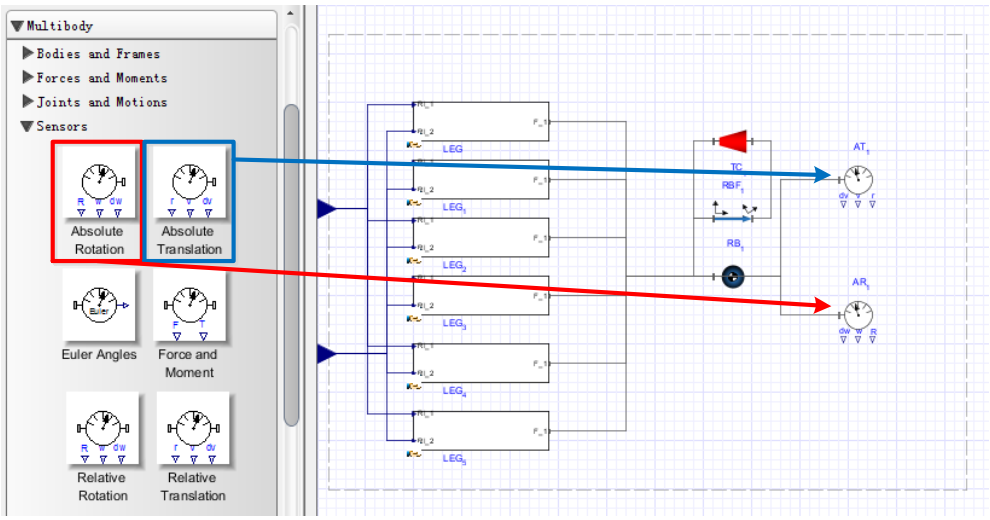


图 12-107　测量上平台位置和转角输出

2. 绝对位移传感器 Absolute Translation 可输出 XYZ 三个方向的位移 r、速度 v 和加速度值 dv。绝对转角传感器 Absolute Rotation 可输出绕 XYZ 三个方向的转角 R、角速度 w、角加速度值 dw，转角的值以 9 个旋转向量的形式输出,其他端口为三个向量形式输出。

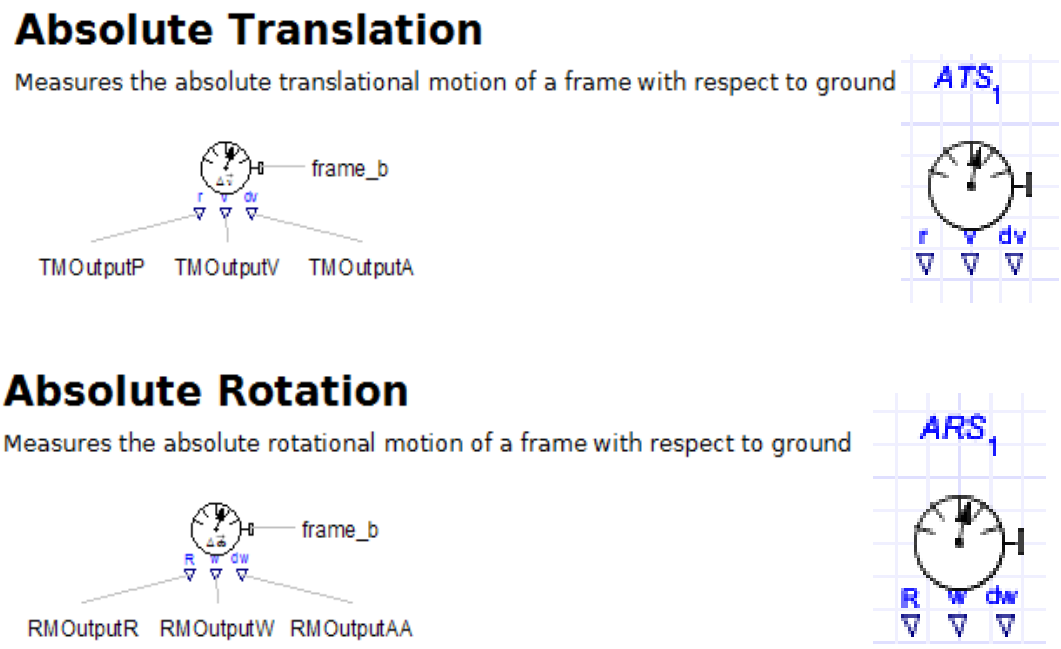


图 12-108　测量元件帮助文档

3. 拉出两个信号分解元件 Real DeMultiplexer，将其连接到 ATS 的位置输出端 r，ARS 速度输出端 w，接着从 Signal Blocks->Common 目录下，拉出三个积分元件 Integrator，按如下图连接。

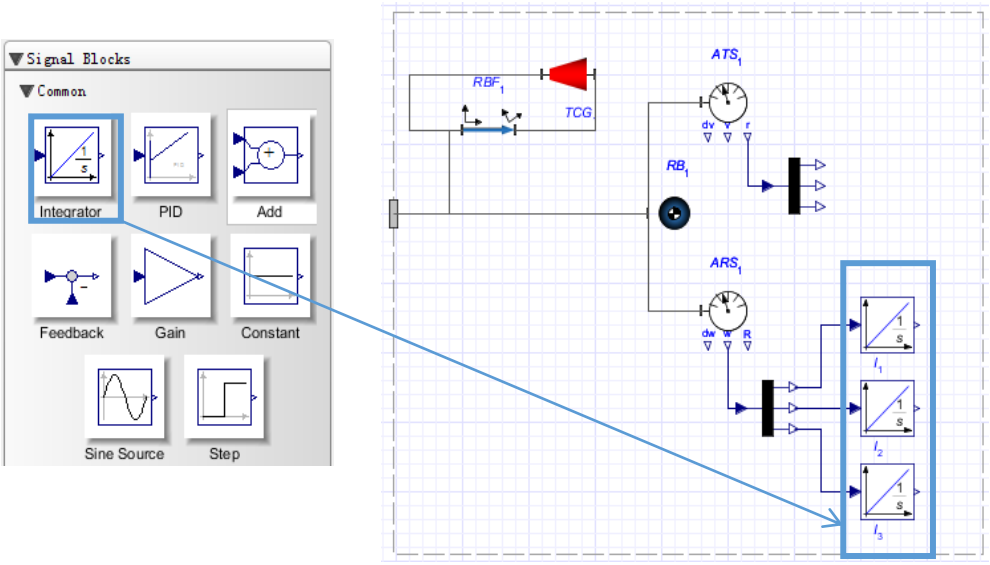


图 12-109　使用积分元件输出上平台转角

4. 将位置测量输出端口依次连接到 Stewart 子系统后侧，如下图所示。

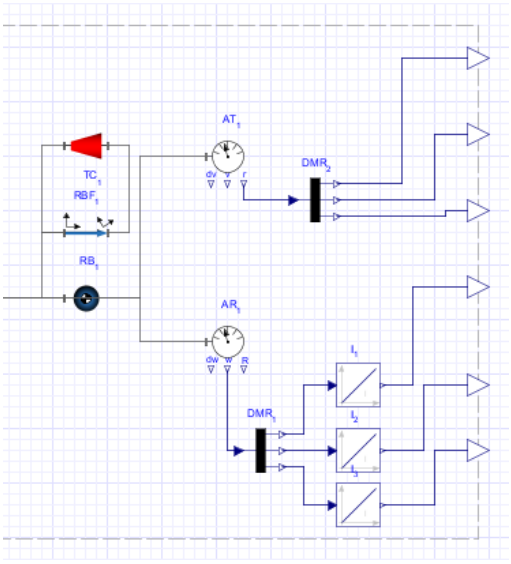


图 12-110　上平台输出端口

5. 回到主目录下，为上平台的输出端口添加探针，并将探针的名字依次修改为 PXout,PYout,PZout,RXout,RYout,RZout 如下图所示。分别用于输出上平台 XYZ 方向的位移量和转角值，运行模型查看平台输出曲线，保存模型。

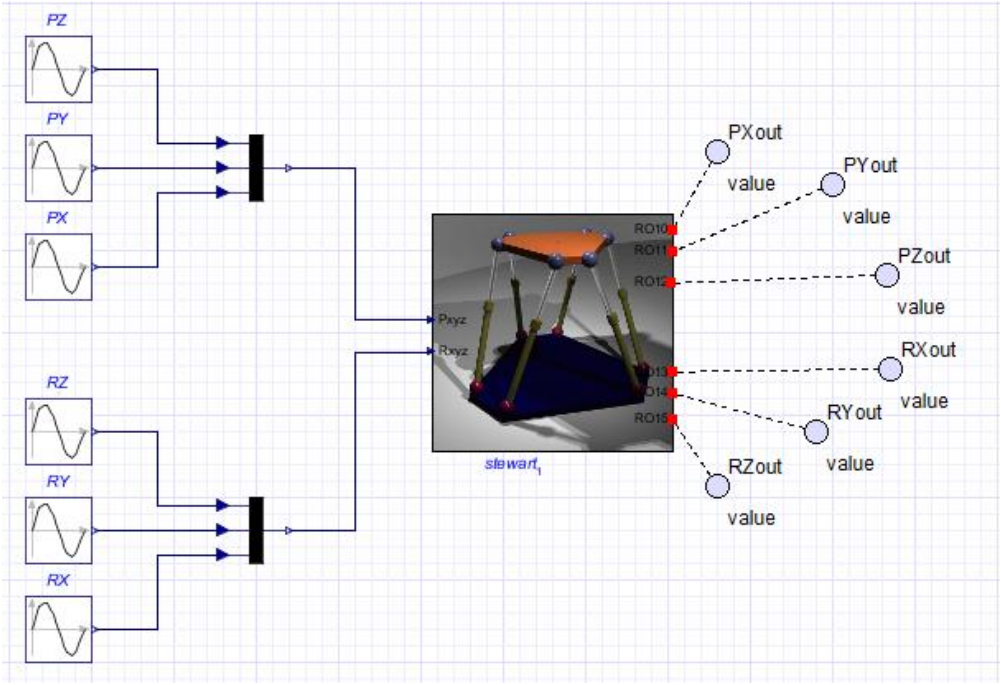


图 12-111　Stewart 跟踪模型

6. 添加探针 probe 用于测量输入信号名称分别为 PXin, PYin, PZin, RXin, RYin, RZin，便于输入输出的效果对比，如下图所示。

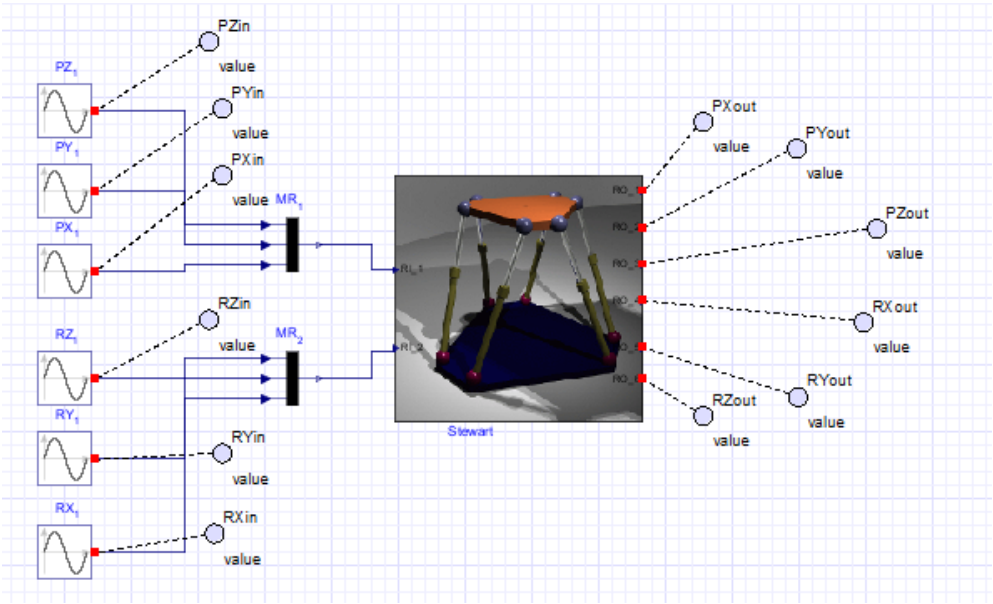


图 12-112　Stewart 输入输出测量

7. 运行模型，查看输入输出对比结果；

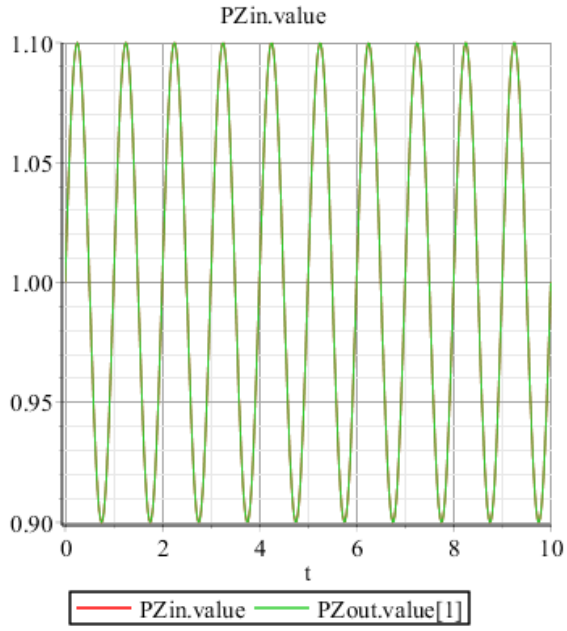


图 12-113　理想跟踪效果

12.4.5 模型 PID 控制

1. 另存模型为 stewartPIDcontrol，进入连杆子系统 LEG，将原有的理想连杆位移驱动 Translational Position 删除，按如下图拉出 PID 元件，Feedback(反馈)元件，Force(力驱动) 元件，Position Sensor(位置传感器)元件。

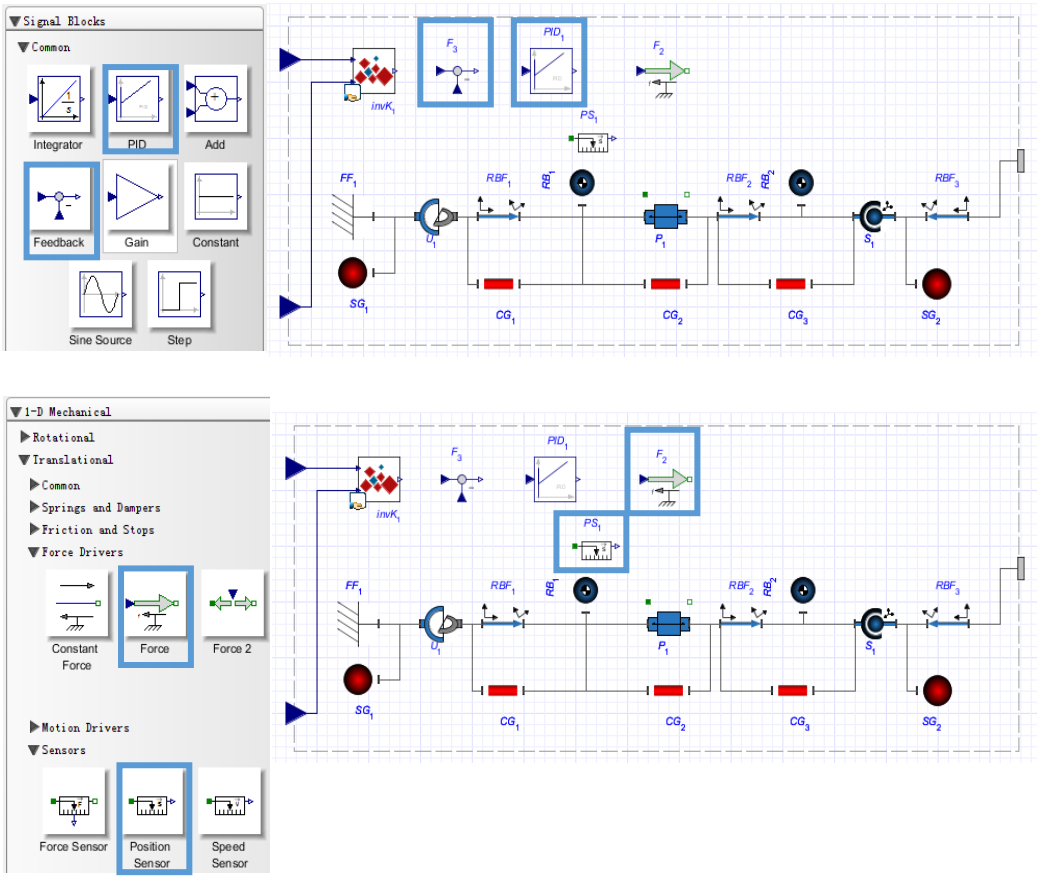


图 12-114　Stewart 模型 PID 稳定控制

2. 按下图连接，并修改 PID 元件的参数 k=30000,T_i=300,T_d=5,N_d=1000(由于 Stewart 模型在正弦信号输入以及重力的影响下，有一个加速度突变，因此将比例系数 K 设的较大)。

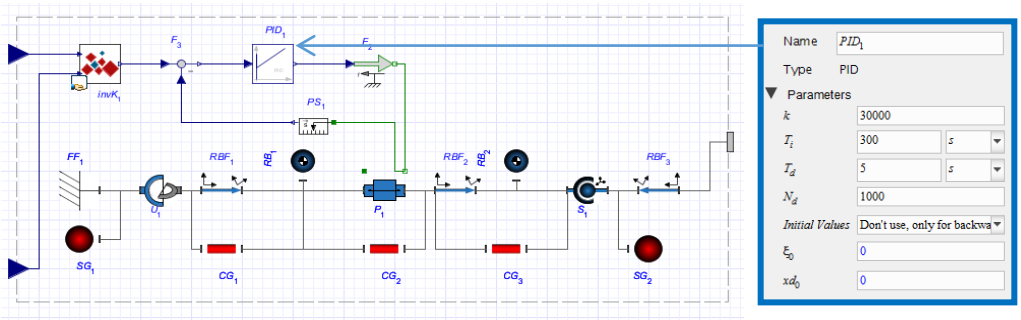


图 12-115　设置 PID 参数

3. 由于平台采用力控制，为保证模型从确定的初始位置开始进行控制，需要将模型的上平台初始值类型修改为 Strictly EnForce，将平台的初始值强制在(0,0,H)处。

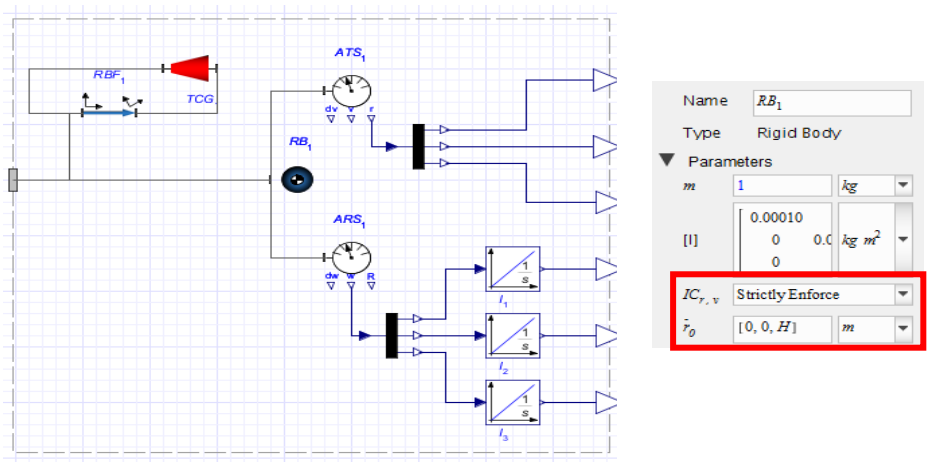


图 12-116　设定上平台初始值类型

4. 运行模型，查看 PID 控制效果。

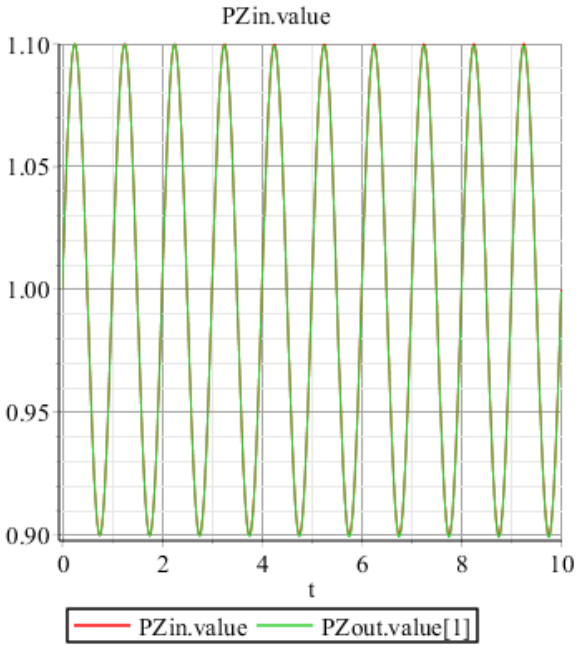


图 12-117　Stewart PID 控制效果

第十三章 雷达定位系统建模

13.1 概要

MapleSim 是一个高性能多领域系统建模和仿真工具，建立在符号—数值混合计算技术基础之上，能够有效地处理工程系统模型（例如复杂多领域系统建模、虚拟样机、和控制系统设计等）开发中涉及的各种复杂数学问题。

模型下载地址：http://www.maplesoft.com.cn/book/13.1.zip

13.1.1 问题描述

飞行器上装载有一个跟踪雷达，在飞行器自由运动时，要求机载雷达方向始终对准地面上的固定目标，以维持更好的通信联系，如下图所示。

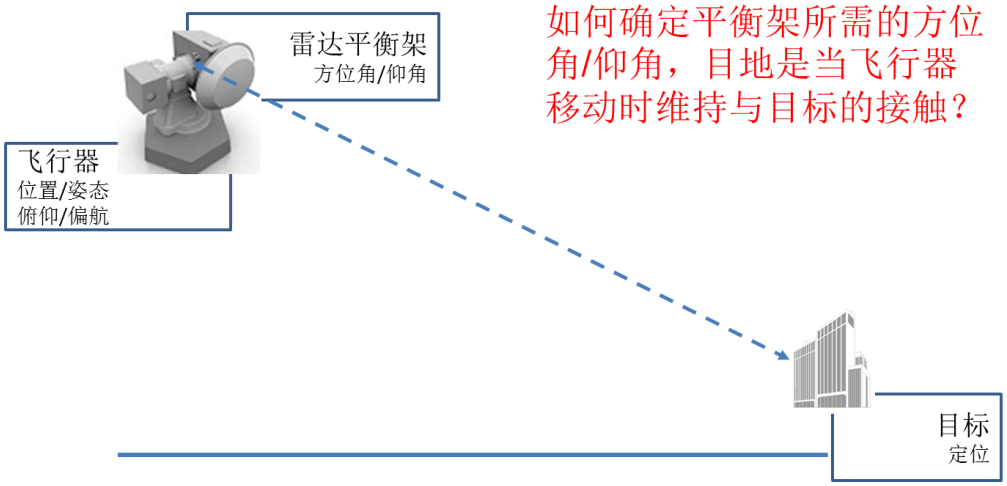


图 13-1 问题描述

13.1.2 问题分析与简化

为了便于分析对模型进行适度简化：

将飞行器的运动（相对于地面）分解为自由的平动和自由的转动两个部分，分别具有 xyz，3 个方向上的自由度；雷达相对于飞行器具有水平、垂直两个方向上的转动，因此将其与信号射线简化成 3 段刚性框架和连接它们的两个方向上的旋转副；设定一个相

对于地面静止的点即为目标。

给于飞行器相对于地面的 6 个自由度任意信号，以代表飞行器的自由运动；给予上述雷达的两个旋转副计算好的信号量，控制射线方向，如图所示。

上述问题实际是，已知物体最后输出的运动，要求输入的控制量。这是一个逆运动学问题。

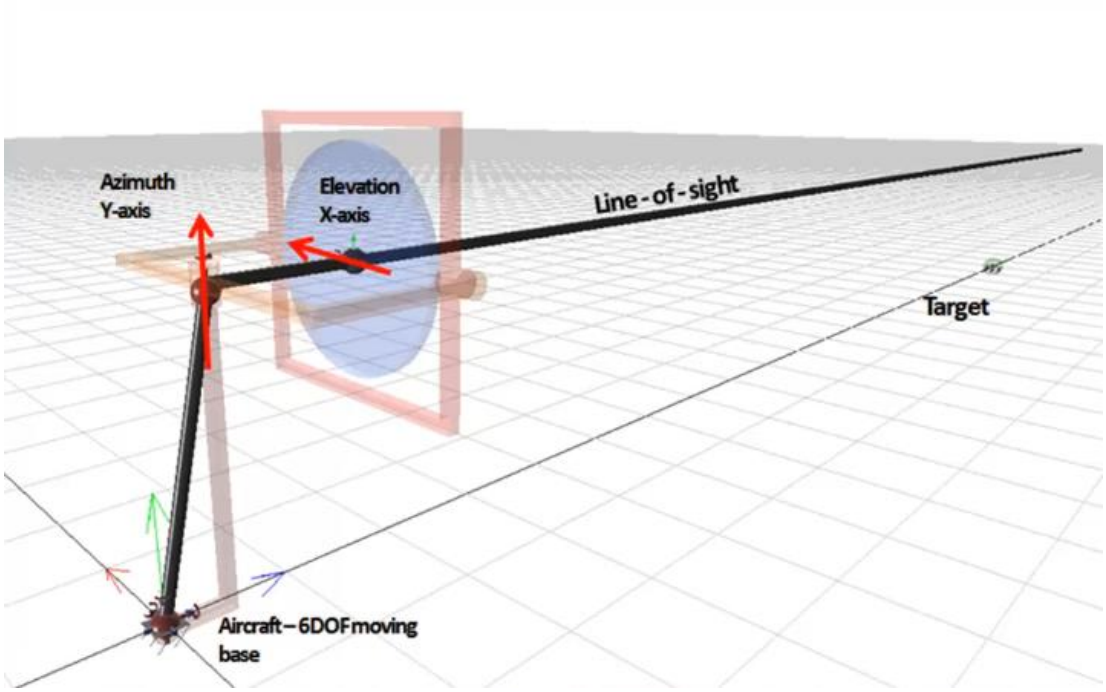


图 13-2　简化的模型

13.2 建立模型

13.2.1 飞行器基点的建立

打开 MapleSim。

首先建立具有 xyz 三方向平动自由度和 xyz 三方向转动自由度的基点，以代表飞行器。

从左侧 **Libraries>Multibody>Joints and Motions** 中拖入以下模型元件，并连接它们，如下图所示。

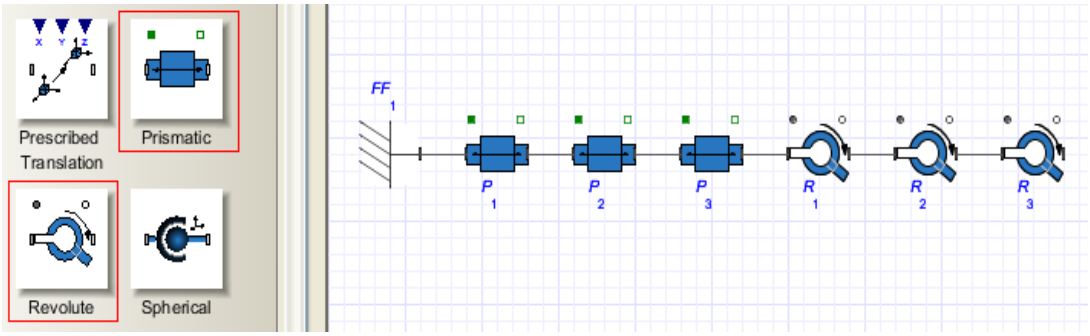


图 13-3　建立具有 6 个自由度的飞行器基点

为了让上述元件分别代表 xyz 不同方向的自由度，分别选中这些元件，然后在右侧的 Inspector 面板内设置相应的参数，即其方向向量。同时，为了满足飞行器可能的各种运动，将初始参数设置为“treat as guess”，即采用 maplesim 程序选择的最优初始参数，以便于后续运动学方程的求解。如下列图所示。

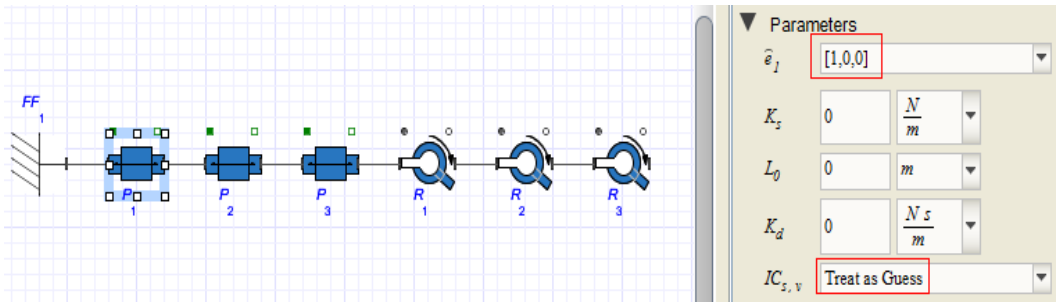


图 13-4　第一个 Prismatic 的参数设置

（用 x 方向滑动副代表 x 方向上的平动）

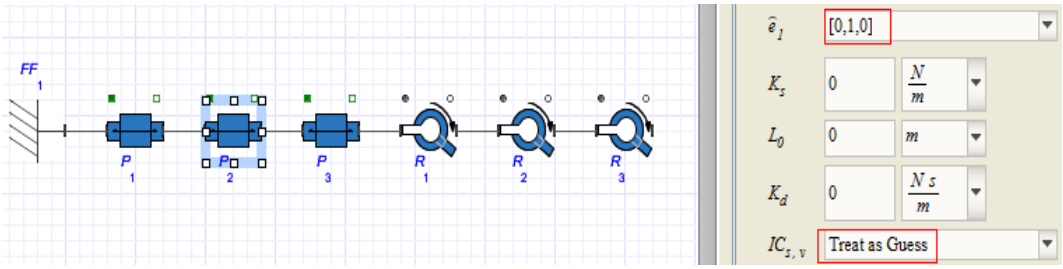


图 13-5　第二个 Prismatic 的参数设置

（用 y 方向滑动副代表 y 方向上的平动）

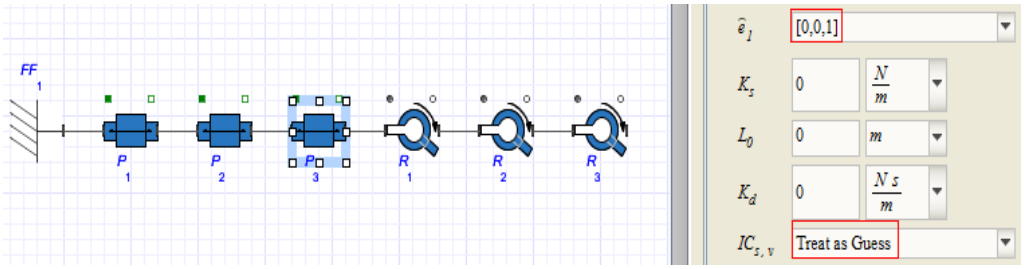


图 13-6　第三个 Prismatic 的参数设置

（用 z 方向的滑动副代表 z 方向上的平动）

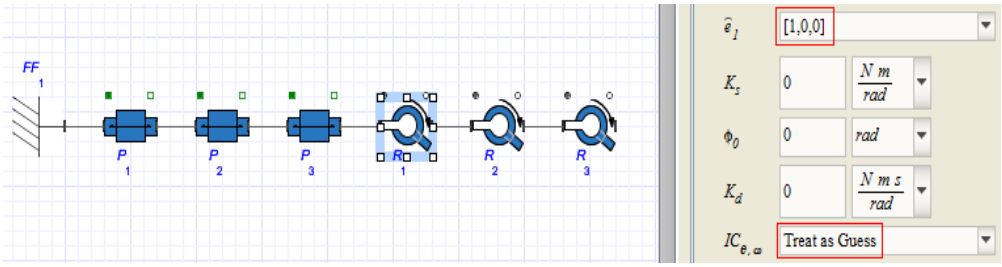


图 13-7　第一个 Revolute 的参数设置

（用 x 方向的旋转副代表 x 方向上的转动）

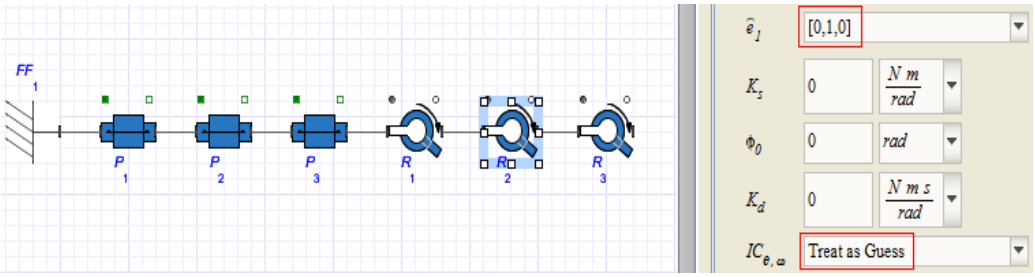


图 13-8　第二个 Revolute 的参数设置

（用 y 方向的旋转副代表 y 方向上的转动）

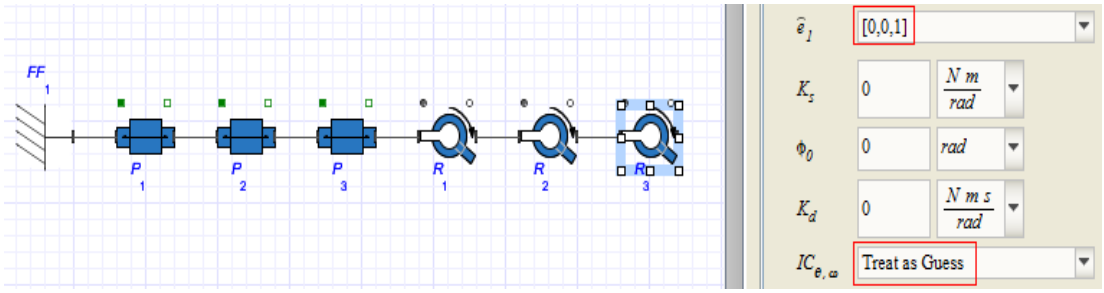


图 13-9　第三个 Revolute 的参数设置

（用 z 方向的旋转副代表 z 方向上的转动）

将上述元件的集合建立一个子系统，以方便后续修改和浏览。

具体操作：按住鼠标左键，选中这 6 个元件（3 个 Prismatic，3 个 Revolute），然后松开鼠标左键，点击鼠标右键（或者直接按组合键 Ctrl+G），选择 Create Subsystem，如图。

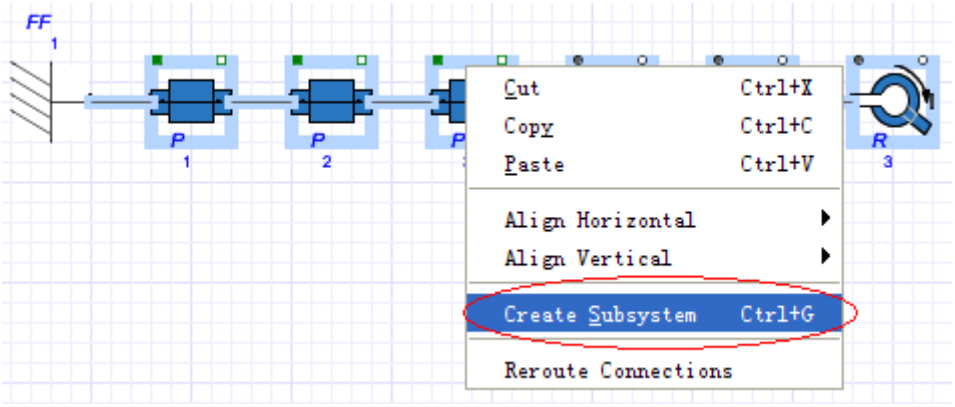


图 13-10　创建一个子系统

在弹出的 Create Subsystem 对话中输入 name：base，将该子系统命名为“base”。

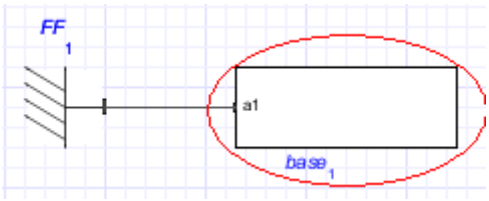


图 13-11　得到的新的子系统

为了得到该子系统与外部其他元件的接口，可以进入该子系统，从想要接出的端口连接出导线连到子系统框架上的绿点上，如下图。

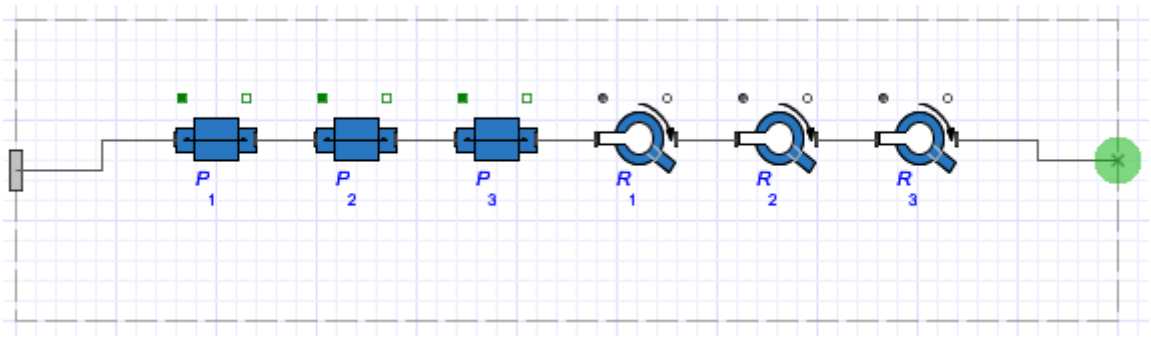


图 13-12　子系统的另一个连接端口

13.2.2 雷达简化模型的建立

为了建立简化后的模型，我们加入 2 个旋转副，用来控制雷达方向，用 2 个刚性的框架将它们连接起来，以形象地显示雷达两个方向角度的变化；并用一个质量块代表雷达的质量，这在后续求解逆运动学和添加 PID 的控制时是必须的。如下图 2.10 所示。

注意：如果需要旋转元件，可以通过选择该元件，然后从右键菜单中选择旋转，也可以使用快捷键实现(ctrl+r，ctrl+f 等)。

从左侧 Libraries>Multibody>Bodies and Frames 中拖入下列元件并连接，如下图。

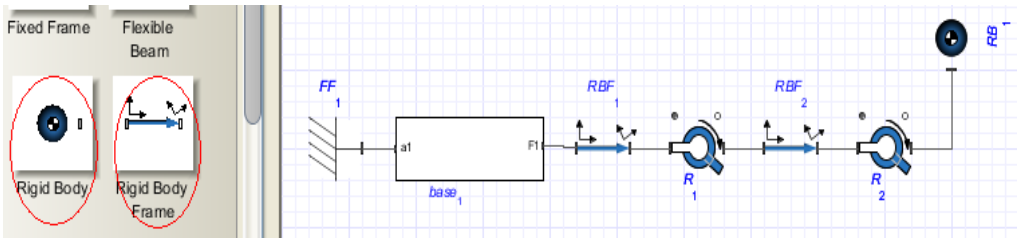


图 13-13　简化的雷达模型及其元件

对于我们要建立的理想模型，我们假设目标在 z 轴上，雷达的初始方向也面向 z 轴，并且在 yz 平面上。2 个旋转副分别控制水平面（xz 平面）和垂直面（yz 平面）的转动。

针对上述假设，定义雷达模型各个元件的参数，如下列图所示。

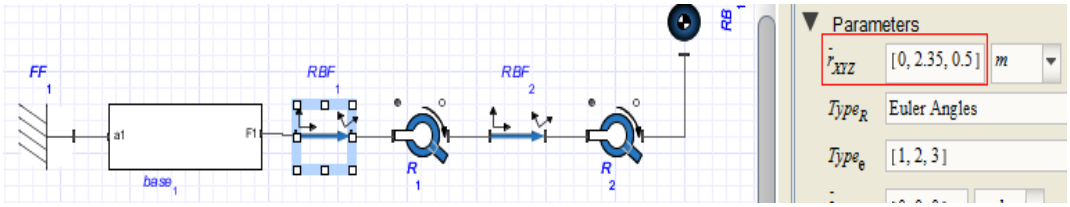


图 13-14　第一个 Rigid Body Frame 的参数设置

第一个 Rigid Body Frame 的矢量坐标为（0，2.35，0.5），在 yz 平面上。以上数值地具体设置，是为了后续添加 CAD 可视化元素时，方便物理元件与视图元件匹配而设定的，具体的数值还可以按情况调整。后续部分参数设置也是同样的原因。

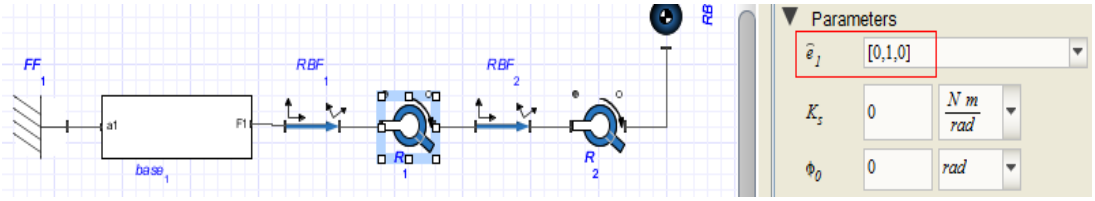


图 13-15　第一个 Revolute 的参数设置

该旋转副控制雷达方向水平方向（xz 平面）转动（方向向量（0,1,0）如上图）。

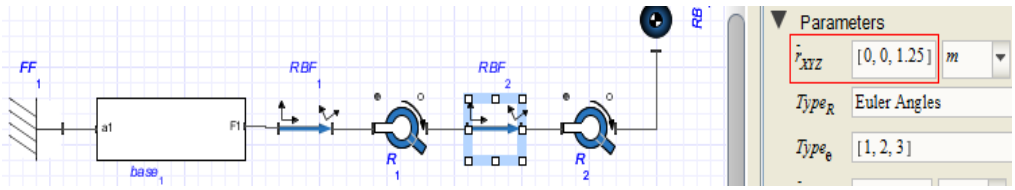


图 13-16　第二个 Rigid Body Frame 的参数设置

该刚性框架相对矢量坐标为（0,0,1.25），连同第一个刚性框架所在的平面为 yz 平面，代表雷达方向的初始方向和位置，如上图。在修改完后，按下 按钮，可以切换到 3-D 图形界面。可以看到，通过对刚性框架的位置矢量的修改，其图形位置发生了相应的变化。

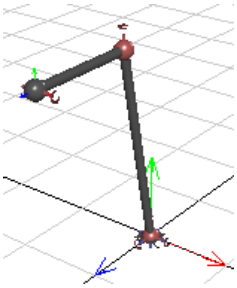


图 13-17　雷达模型在图形界面的表达

选中编辑界面的某个元件或者选中图形界面的某个图形，都可以对应到相应的图形或者元件，如下图所示，选中了第二个刚性框架，图形界面中对应的连杆颜色发生了变化，表示该连杆为第二个刚性框架这个元件。

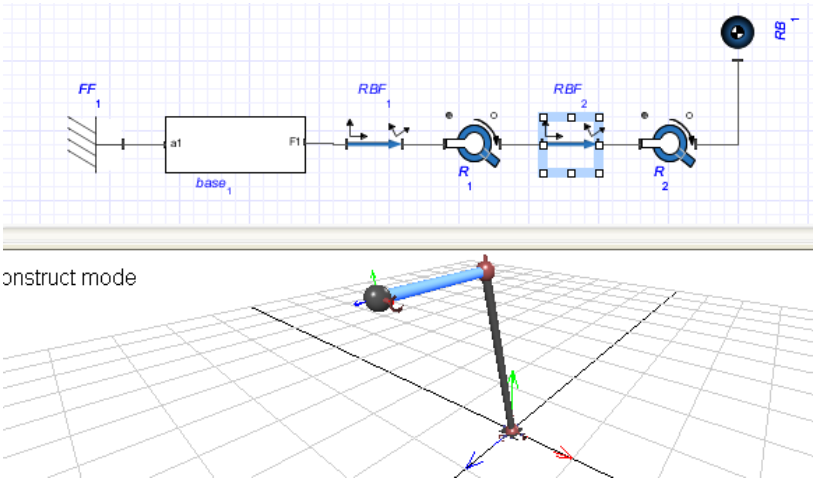


图 13-18　元件（蓝框选中）与对应的图形（蓝色连杆）表达

继续进行参数设置：

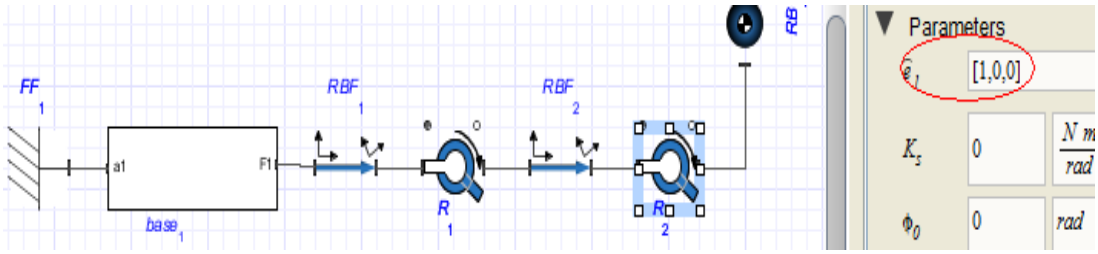


图 13-19　第二个 Revolute 的参数设置

该旋转副控制雷达方向垂直方向（yz 平面）转动（方向向量（1,0,0）如上图）。

质量块 Rigid Body 的参数定义对于结果无太大影响，可以使用默认的各个参数，也可以针对实际需要进行修改。

13.2.3 位置约束的建立

为了求解输入的控制量，求解这个逆运动学问题，我们需要约束输出的物体的运动。在此例中，需要建立雷达与目标之间的约束。最终求解得到的是雷达上 2 个旋转副的角度控制量，所以应当建立射线（即第三段刚性框架）与目标点之间的全角度约束，可以采用的球面副表示；同时雷达与目标点之间还有距离的变化，可以用滑动副表示。

如下图加入 3 个元件，并连接：

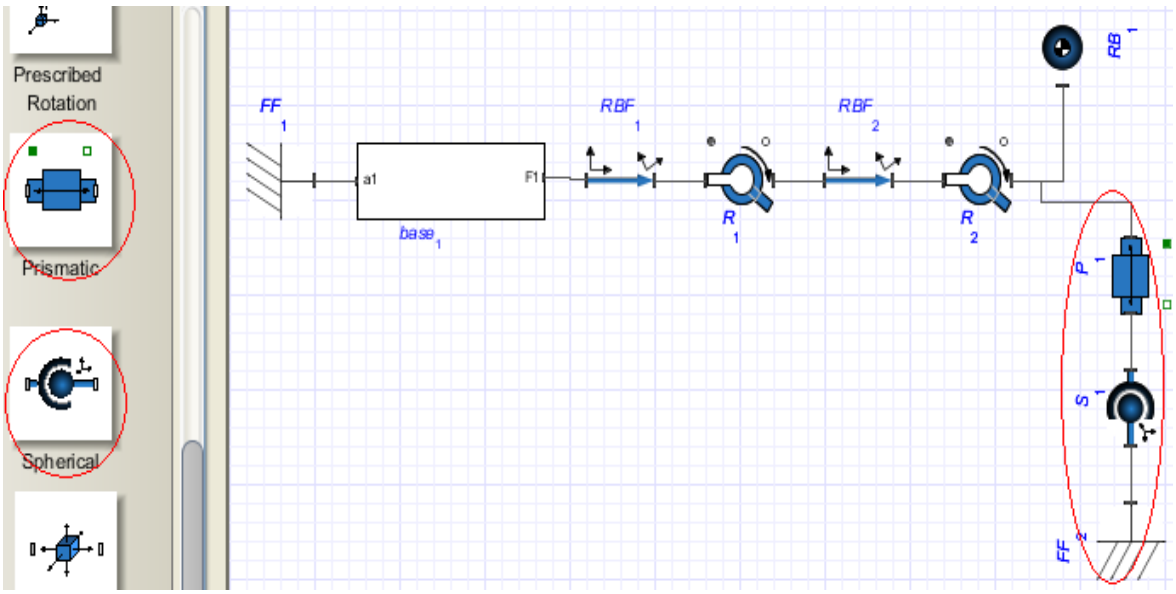


图 13-20　位置约束模型及其元件

各个元件具体参数设定如下：

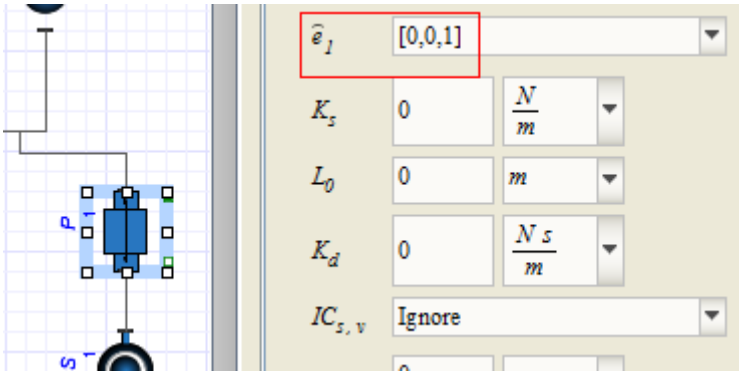


图 13-21　位置约束中滑动副（代表距离约束）的参数设定

因为目标点的初始相对位置在雷达的 z 方向上，射线（即第三段刚性框架）的长短变化也在相对坐标 z 方向上，所以设定该滑动副的方向向量设定为（0,0,1），如上图。

球面副的各个参数采用默认的设置即可，无需改动。

根据我们开始的假设，目标点位置在 z 轴上，所以设定 Fixed Frame$_2$ 的位置参数为（0,0,20），如下图 2.18 所示。

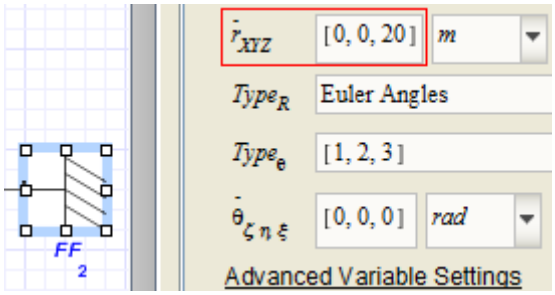


图 13-22　目标点 Fixed Frame$_2$ 参数设置

许多情况下，我们可能需要用到全局参数或者变量。下面以上述目标点 Fixed Frame$_2$ 的参数设置为例，用全局参数的方式设置。

具体操作：点击左上工具栏上的视图按钮，切换到参数定义视图，如下图所示。

图 13-23　参数定义视图

添加参数 Tx，Ty，Tz，并分别取值为 0,0,20，如下图。

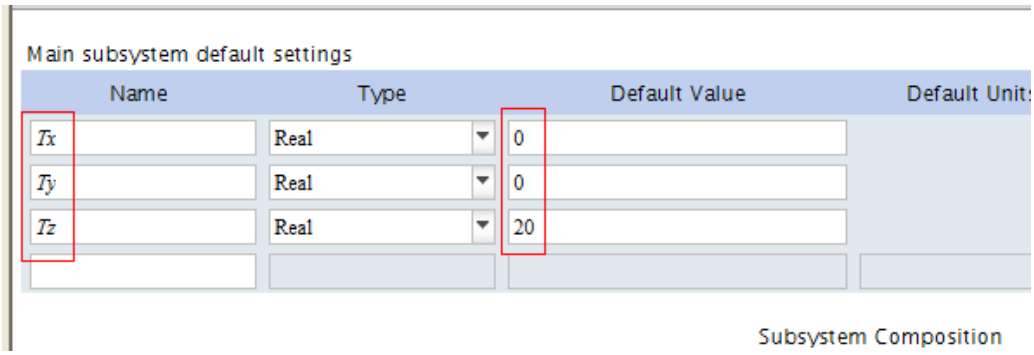


图 13-24　添加 Tx、Ty、Tz 三个参数

参数定义完成后，点击  视图按钮，返回到模型框图。选中 Fixed Frame2 元件，修改位置矢量参数为（Tx，Ty，Tz），如此 Fixed Frame2 的位置矢量参数即设定为了（0,0,20），同时也将该参数提取为全局参数，可以在其他需要的地方应用上。

13.3 建立逆运动学的自定义模块

上述模型仅仅建立了机构间的约束，可以得到系统的运动约束方程，而不是用于仿真。为了进行仿真，需要加入信号激励，以模拟飞行器的运动；同时输入计算得到的旋转副控制量，用雷达追踪目标，以观察仿真结果。

13.3.1 提取系统的运动约束方程

点击左侧项目面板上的 Add Apps and Templates 按钮（  ），选择列表中的 Worksheet、，打开 Maple 界面，如下列图所示。

图 13-25 创建多体运动学的 maple 模板

在 Maple 界面中，如下图：

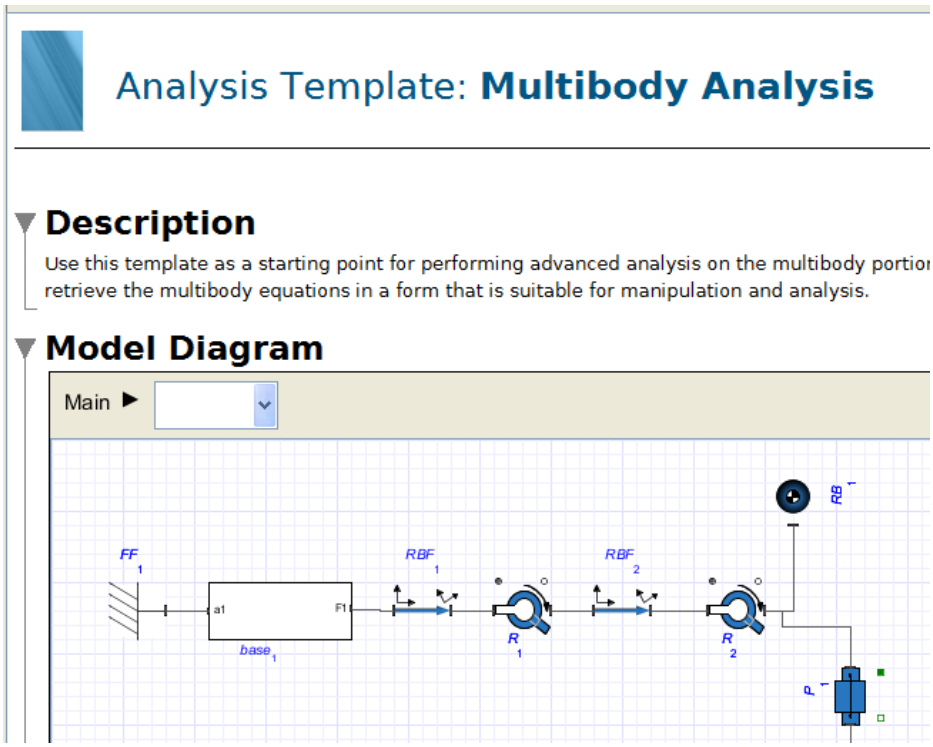


图 13-26　跟踪雷达模型的 Multibody Analysis 模板

该模板，是 Maplesim 自动生成的用于多体动力学分析的模板，有许多方程或命令行，可以求解得到该系统中的许多动力学方程或矩阵，用户在对 Maple 有一定熟悉的情况下，也可以自己添加、修改方程、命令行，得到自己想要的动力学方程等数学信息。

依次选择各个命令行，按回车运行这些命令，或直接点击 !!! 全部运行。

注意：命令语句有用冒号，表示隐藏输出。用分号，表示显示输出。

运行如下程序：

```
mModel := MapleSim:-LinkModel();

MB := mModel:-GetMultibody(simplify = true);
```

生成并显示系统的控制方程，如下图。

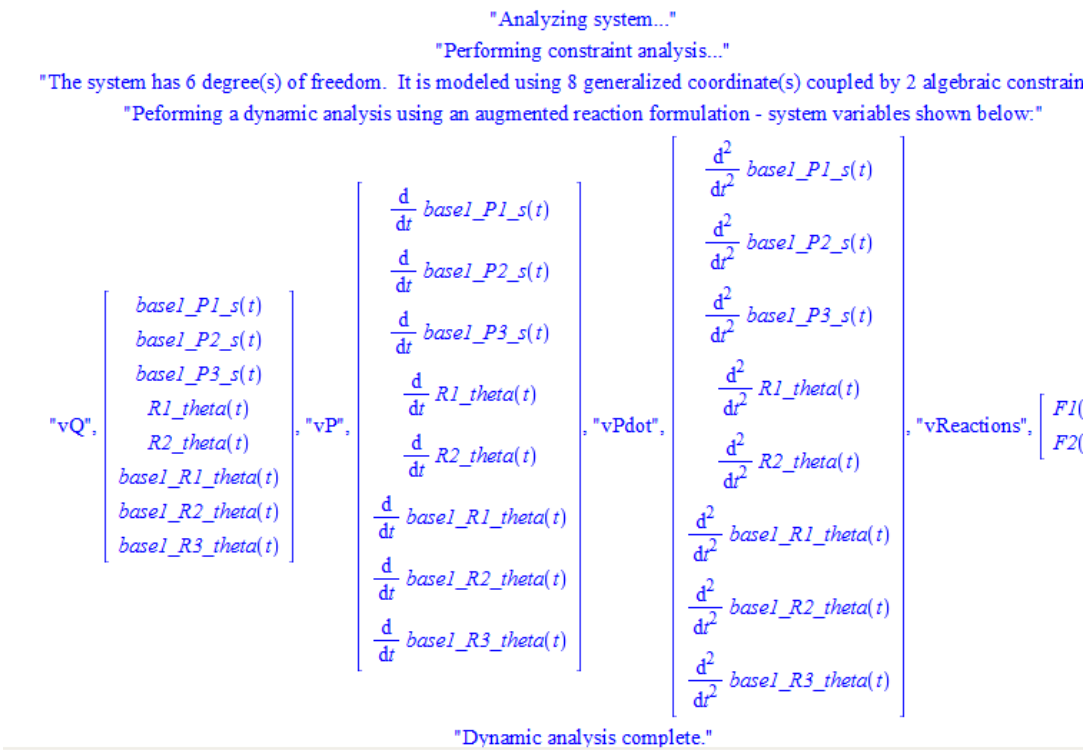


图 13-27　得到的系统控制方程

将之前输入的程序删除，　并输入以下程序：

```
mModel := MapleSim:-LinkModel();
```

```
vPosCons:=MB:-GetPosCons();
```

该命令得到了系统的运动方程的位置矩阵，显示并观察该矩阵。

该矩阵有 2 个元素，分别为 R1（即第一个旋转副）的角度表达式和 R2（即第二个旋转副）的角度表达式。

在上述程序后加入下面的命令，求解旋转副水平和垂直角度的运动学约束方程：

```
th1:=solve(MB:-GetPosCons()[1],{R1_theta(t)});
```

```
th2:=solve(MB:-GetPosCons()[2],{R2_theta(t)});
```

13.3.2 建立自定义组件

通过上述操作，已经得到了 2 个控制旋转副的运动学约束方程，下一步，需要提取这两个方程，用来计算需要给出的 2 个旋转副的控制量。

返回到 MapleSim 界面中，点击左侧项目面板上的 Add Apps and Templates 按钮（  ），选择打开 Custom Component，如下图。

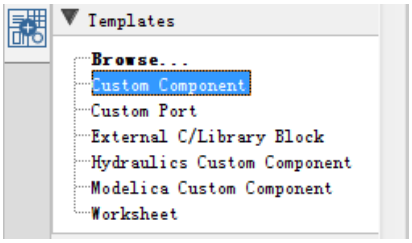


图 13-28　创建用户自定义组件

将得到的运动学约束方程加入到该用户自定义组件中，用来计算 2 个控制量。

具体操作：

删除 eq 中的方程，保留方括号 $eq := [\,]$ 。

返回到 Multibody Analysis 界面。复制并粘帖两个解到 Custom Component 中 eq 后的方括号中，两个方程之间用逗号（,）分隔，如下图。

```
> th1 := solve(MB:-GetPosCons( )[1], {R1_theta(t)})
th1 := {R1_theta(t) = arctan((2 (base1_P2_s(t) sin(base1_R2_theta(t)) sin(
    − base1_P3_s(t) sin(base1_R2_theta(t)) cos(base1_R1_theta(t)) cos(
    + 20 cos(base1_R1_theta(t)) cos(base1_R3_theta(t)) sin(base1_R2_th
    + sin(base1_R3_theta(t)) base1_P3_s(t) sin(base1_R1_theta(t)) + sin
    − 20 sin(base1_R1_theta(t)) sin(base1_R3_theta(t))))/(−40 cos(base
    + 2 cos(base1_R2_theta(t)) base1_P3_s(t) cos(base1_R1_theta(t)) +
    − 2 cos(base1_R2_theta(t)) base1_P2_s(t) sin(base1_R1_theta(t))))}
```

A、复制 R1 的运动约束方程

```
> th2 := solve(MB:-GetPosCons( )[2], {R2_theta(t)})
th2 := {R2_theta(t) = arctan((1/5) (400 cos(base1_R1_the
    + 400 sin(base1_R1_theta(t)) cos(base1_R3_theta(
    + 20 base1_P2_s(t) sin(base1_R2_theta(t)) sin(ba
    − 20 base1_P3_s(t) cos(base1_R1_theta(t)) sin(ba
    + 20 base1_P1_s(t) cos(base1_R2_theta(t)) sin(ba
    − 20 base1_P3_s(t) cos(base1_R3_theta(t)) sin(ba
    − 80 cos(R1_theta(t)) cos(base1_R1_theta(t)) cos(
    + 4 base1_P2_s(t) sin(base1_R2_theta(t)) sin(R1_
    − 4 base1_P3_s(t) sin(base1_R2_theta(t)) sin(R1_
    + 80 cos(base1_R1_theta(t)) sin(base1_R2_theta(t
    + 4 sin(base1_R3_theta(t)) base1_P3_s(t) sin(R1_
    + 4 cos(base1_R2_theta(t)) base1_P3_s(t) cos(bas
    + 4 base1_P1_s(t) cos(base1_R2_theta(t)) sin(R1_
    + 4 sin(base1_R3_theta(t)) base1_P2_s(t) cos(bas
    − 4 cos(base1_R2_theta(t)) base1_P2_s(t) cos(R1_
    + 2 cos(R1_theta(t))))/(−40 cos(base1_R1_the
    + 400 sin(base1_R1_theta(t)) sin(base1_R3_theta(
    + 20 base1_P2_s(t) cos(base1_R1_theta(t)) sin(ba
    + 20 base1_P3_s(t) sin(base1_R1_theta(t)) sin(ba
    + 20 base1_P1_s(t) sin(base1_R2_theta(t)) sin(ba
    − 20 base1_P3_s(t) sin(base1_R3_theta(t)) sin(ba
    − 80 sin(R1_theta(t)) cos(base1_R1_theta(t)) cos(
    + 4 base1_P2_s(t) cos(base1_R2_theta(t)) sin(R1_
    − 4 base1_P3_s(t) cos(base1_R2_theta(t)) sin(R1_
    − 80 cos(base1_R1_theta(t)) sin(base1_R2_theta(t
    + 4 cos(base1_R3_theta(t)) base1_P3_s(t) sin(R1_
    − 4 sin(base1_R2_theta(t)) base1_P3_s(t) cos(bas
    + 4 base1_P1_s(t) sin(base1_R2_theta(t)) sin(R1_
    + 4 cos(base1_R3_theta(t)) base1_P2_s(t) cos(bas
    − 4 sin(base1_R2_theta(t)) base1_P2_s(t) cos(R1_
    + 2 sin(R1_theta(t))))}
```

B、复制 R2 的运动学约束方程

$$\text{eq} := \Big[R1\_theta(t) = \arctan\big((2\,(base1\_P2\_s(t)\sin(base1\_R2\_theta(t)))\sin(base1\_R1\_theta(t)))\cos(base1\_R3\_theta(t)) - base1\_P3\_s(t)\sin(base1\_R2\_theta(t))\cos(base1\_R1\_theta(t))\cos(base1\_R3\_theta(t)) + 20\cos(base1\_R1\_theta(t))\cos(base1\_R3\_theta(t))\sin(base1\_R2\_theta(t)) + base1\_P1\_s(t)\cos(base1\_R2\_theta(+ \sin(base1\_R3\_theta(t)))\,base1\_P3\_s(t)\sin(base1\_R1\_theta(t)) + \sin(base1\_R3\_theta(t))\,base1\_P2\_s(t)\cos(base. - 20\sin(base1\_R1\_theta(t)))\sin(base1\_R3\_theta(t))))\big/(-40\cos(base1\_R1\_theta(t)))\cos(base1\_R2\_theta(t))) + 1 + 2\cos(base1\_R2\_theta(t))\,base1\_P3\_s(t)\cos(base1\_R1\_theta(t)) + 2\,base1\_P1\_s(t)\sin(base1\_R2\_theta(t)) - 2\cos(base1\_R2\_theta(t))\,base1\_P2\_s(t)\sin(base1\_R1\_theta(t)))), R2\_theta(t) = \arctan\Big(\frac{1}{5}(400\cos(base1\_R1\_theta(t)))\sin(base1\_R2\_theta(t)))\sin(base1\_R3\_theta(t))) + 400\sin(base1\_R1\_theta$$

C、粘贴到 Custom Component 模板 eq 中

图 13-29　提取 2 个约束方程到用户组件的方程中

拷贝完成后，按回车键，执行 eq 这一行的语句。

将参数和初始条件全设置为空，并运行，如下图。

$$\text{params} := [\,]$$

$$[\,]$$

$$\text{initialconditions} := [\,]$$

$$[\,]$$

图 13-30　用户组件中参数和初始条件设置

对于设定了全局参数的，须如下设定参数和初始条件。

$$\text{params} := [Tx = 0, Ty = 0, Tz = 20]$$

$$[Tx = 0, Ty = 0, Tz = 20]$$

$$\text{initialconditions} := [\,]$$

$$[\,]$$

图 13-31　定义了全局参数的情况下设定用户自定义组件参数和初始条件

通过以上操作，定义了用户自定义组件的系统方程、参数和初始条件。接下来定义该组件的物理端口。该模型中，每组飞行器的 6 个自由度信号对应 2 个旋转副的控制信号，所以需要定义 6 个输入信号和 2 个输出信号，例如下图所示。

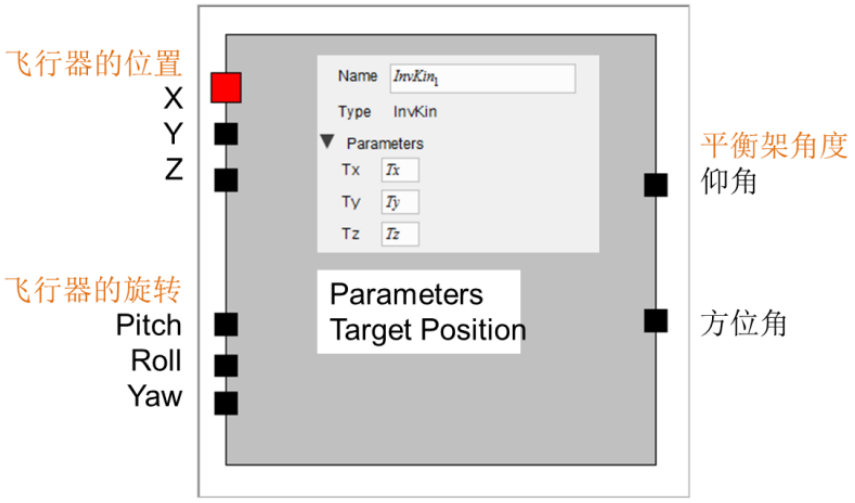


图 13-32　自定义组件的端口涵义

具体操作：点击 Configuration 中的 Ports，进行用户组件的端口定义。定义端口，首先点击**清除所有端口**按钮，清除所有以前的输入和输出端口。然后再点击**添加端口**按钮，添加 8 个端口（端口之间最好尽量分开，并区分输出和输入，如图）。

按下图的顺序依次选择 8 个端口，分别定义端口内容，如图所示。

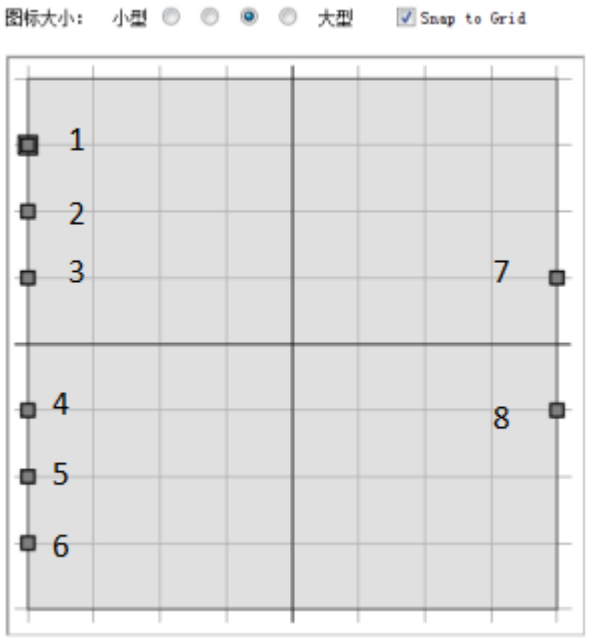
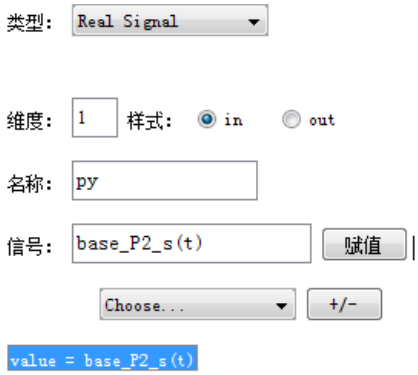


图 13-33　依次选择并定义 8 个端口（左 6 个为输入，右 2 个为输出）的内容

各个端口的具体内容如下：

A）1 号端口的内容设置（x 方向滑动副信号输入）



B）2 号端口的内容设置（y 方向滑动副信号输入）

C）3 号端口的内容设置（z 方向滑动副信号输入）

D）4 号端口的内容设置（x 方向旋转副信号输入）

E）5 号端口的内容设置（y 方向旋转副信号输入）

F）6 号端口的内容设置（z 方向旋转副信号输入）

G）7 号端口的内容设置（R1 旋转副控制量输出）

H）8 号端口的内容设置（R2 旋转副控制量输出）

图 13-34　依次设定 8 个端口的具体内容

在 Custom Component 模板中，可以修改该组件名称，如改为：InvKin 等等。

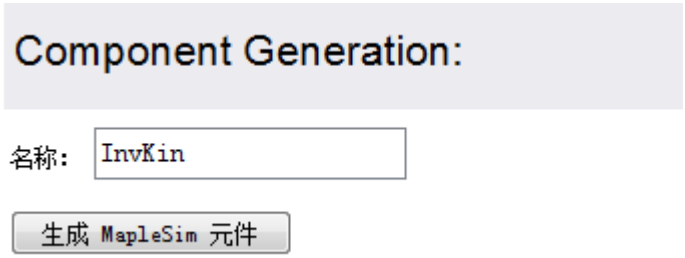


图 13-35　生成 MapleSim 元件

完成上述操作后，继续下拉该模板，点击**生成 MapleSim 元件**，生成以上操作建立的用户自定义组件。

生成好的 MapleSim 自定义组件出现在左侧面板中，可以如其他元件一样使用。

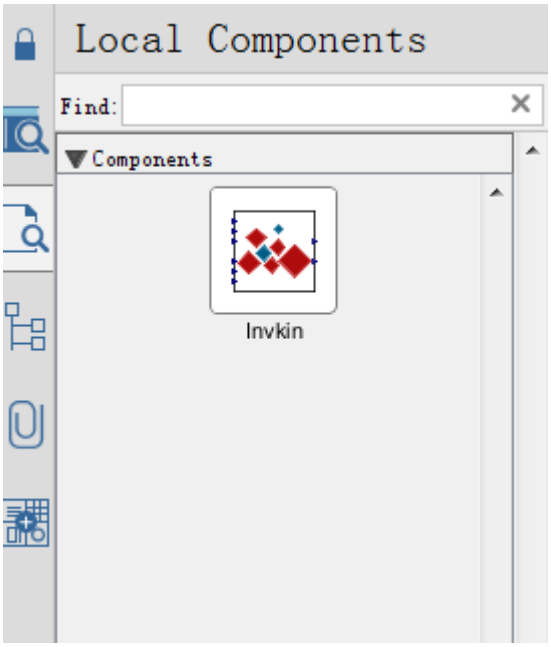


图 13-36　MapleSim 中生成的用户组件图标

13.4 整体系统组建

在原有模型中加入建立的用户组件 InvKin，并添加飞行器运动的信号，一方面该信号输入到用户组件 InvKin 中，通过计算得到 2 个旋转副的控制量输出；另一方面该信号转化为机械信号（或运动信号），输出给飞行器基点，使其模拟飞行器的运动。再添加各种信号连接、转化的元件，完成整个系统的组建。

13.4.1 理想控制系统

定义 3 个正弦信号，控制飞行器的位置 xyz 三方向的平动，并建立成一个子系统，如下图（正弦信号：Signal Blocks>Common>Sine Source，信号合并元件:Signal Blocks>Routing>Multiplexers>Real Multiplexer）.

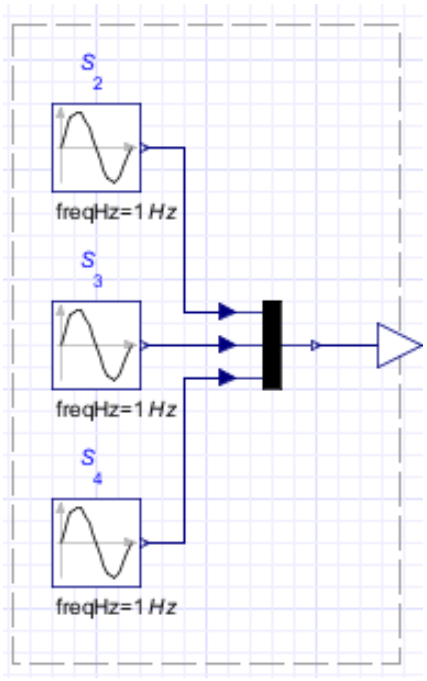
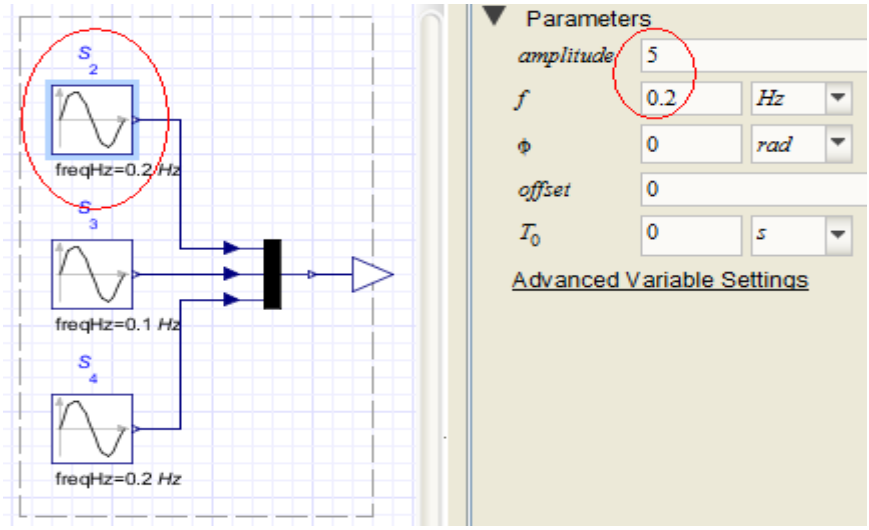
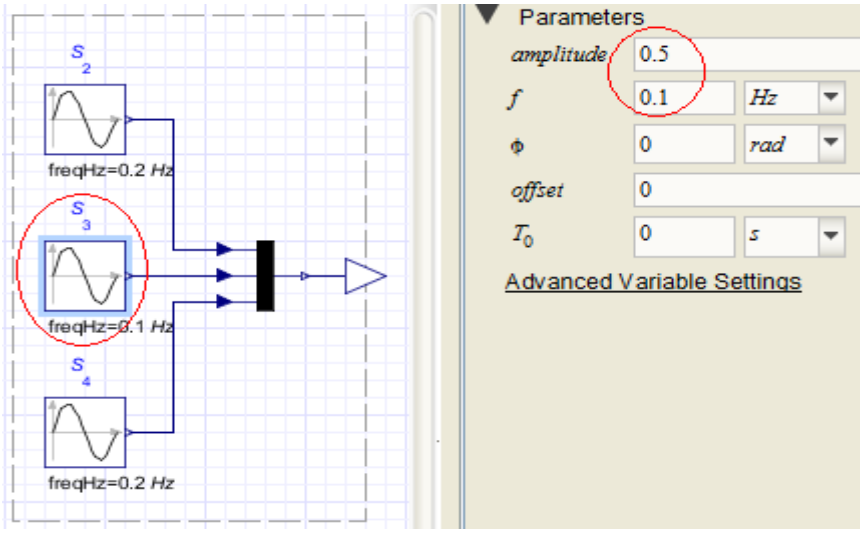


图 13-37　飞行器的位置信号子系统

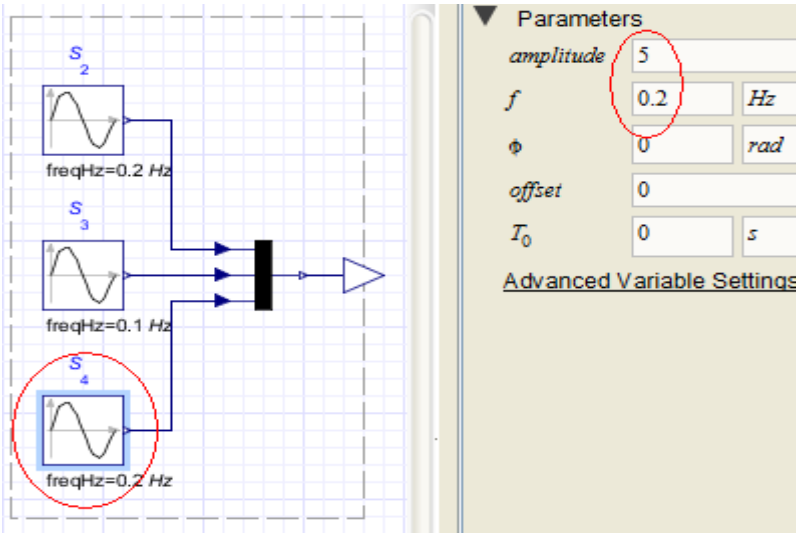
对于以上 3 个信号的参数，可以随意设置，在此假定飞行器周期运动，具体参数设置如图所示。



d）x 方向平动信号参数设置（幅值 5、频率 0.2Hz）



e）y 方向平动信号参数设置（幅值 0.5、频率 0.1Hz）



f）z 方向平动信号参数设置（幅值 5、频率 0.2Hz）

图 13-38　飞行器位置信号的参数设定

将 InvKin 元件拖入到模型框图中，使用方式与 MapleSim 内置的元件相同。再加入一个信号分离元件（Signal Block>Routing>Demultiplexers>Real Demultiplexer），并依此连接信号与端口，如下图。

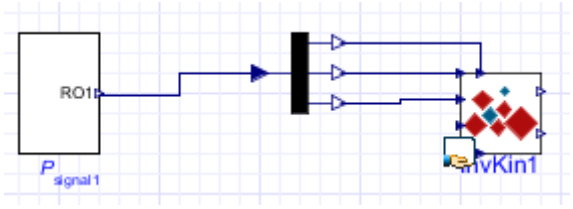


图 13-39　连接飞行器位置信号子系统与用户组件 InvKin

进入飞行器基点 base 子系统，添加一个信号分离元件，同时为了将该信号转化为驱动运动副的机械信号，须添加平动位置驱动（1-D Mechanical>Translational>Motion Drivers>Position），如下图。

如图连接各个元件，注意信号的顺序。

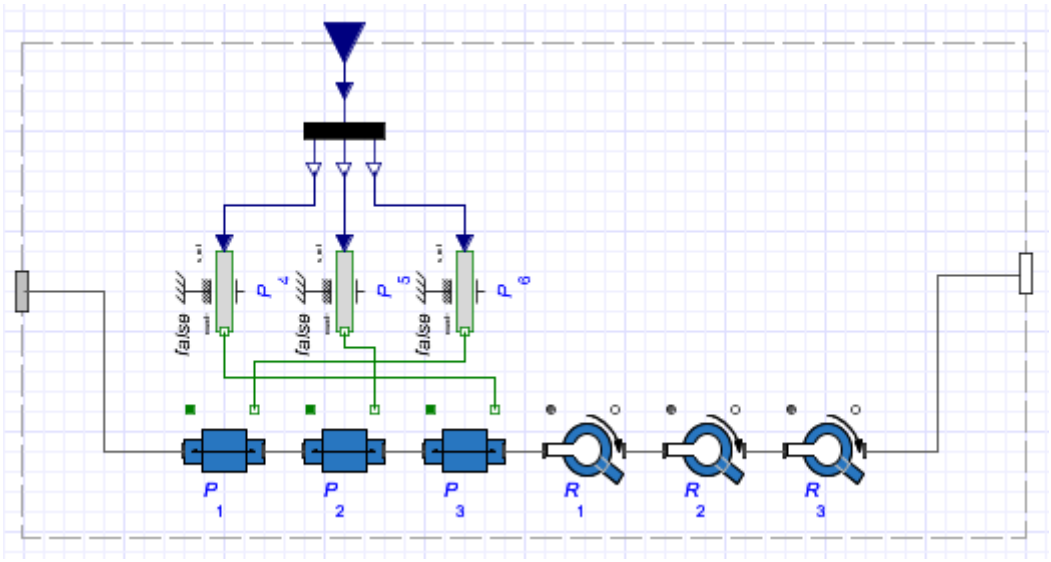
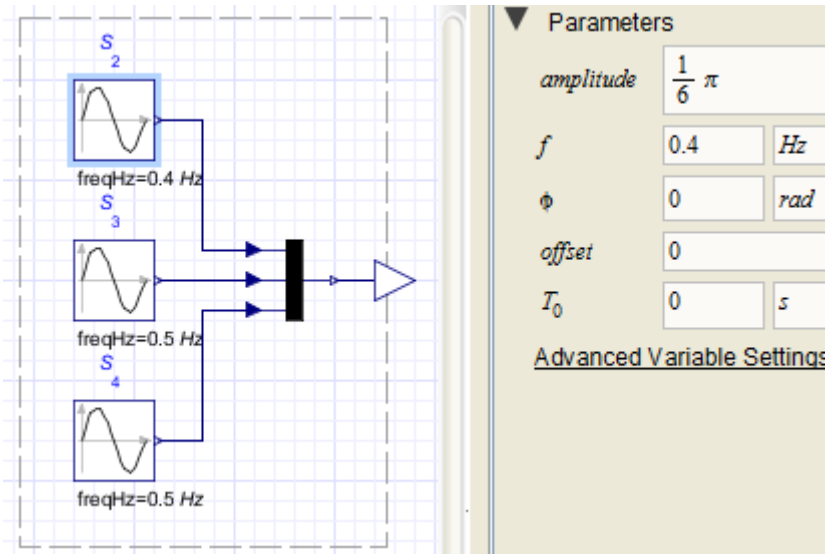
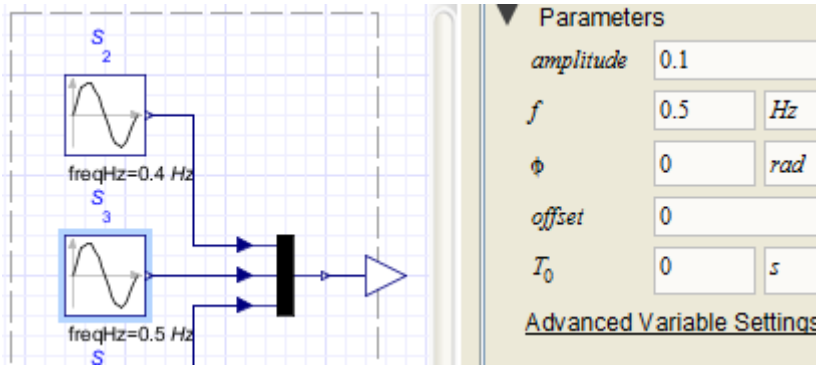


图 13-40　在基点 base 子系统中添加平动位置驱动（注意信号与元件的对应）

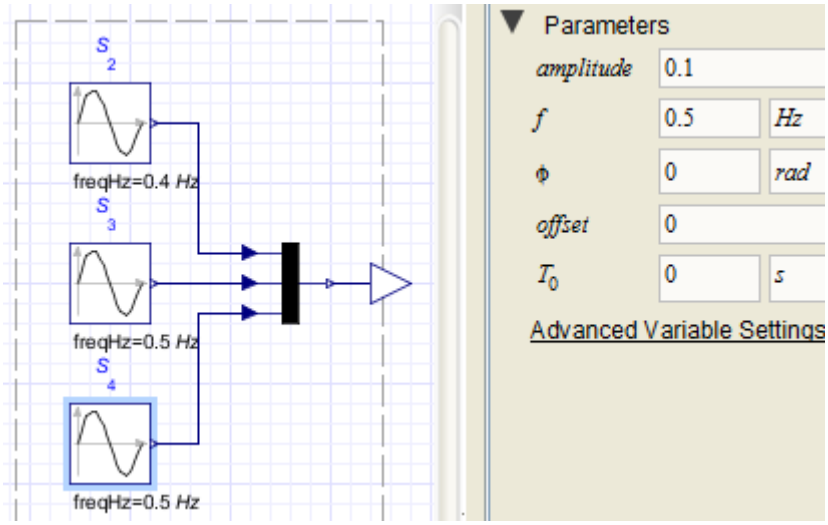
同理创建飞行器旋转的信号子系统，并通过上述类似的操作，建立连接（注意：连接旋转副的运动驱动应为旋转类的运动驱动 1-D Mechanical>Rotational>Motion Drivers>Translational Position）。具体情况和参数设定如下列图所示。



A）飞行器旋转信号 x 方向的参数设置（幅值(1/6)*Pi，频率 0.4Hz）



B）飞行器旋转信号 y 方向的参数设置（幅值 0.1，频率 0.5Hz）



C）飞行器旋转信号 z 方向的参数设置（幅值 0.1，频率 0.5Hz）

图 13-41　飞行器旋转信号子系统各个元件参数的设置

飞行器基点 base 子系统内的信号转化和连接：

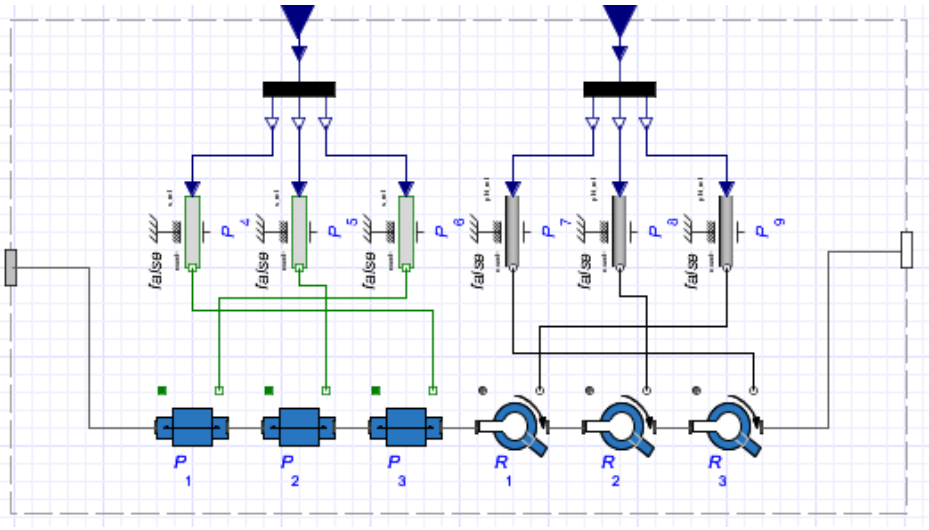


图 13-42　基点 base 子系统内的元件及其连接（注意信号与元件的对应）

注意：连接旋转副的运动驱动应为旋转类的运动驱动

1-D Mechanic>Rotational>Motion Drivers>Translational Position。

以上都完成后，如下图连接。

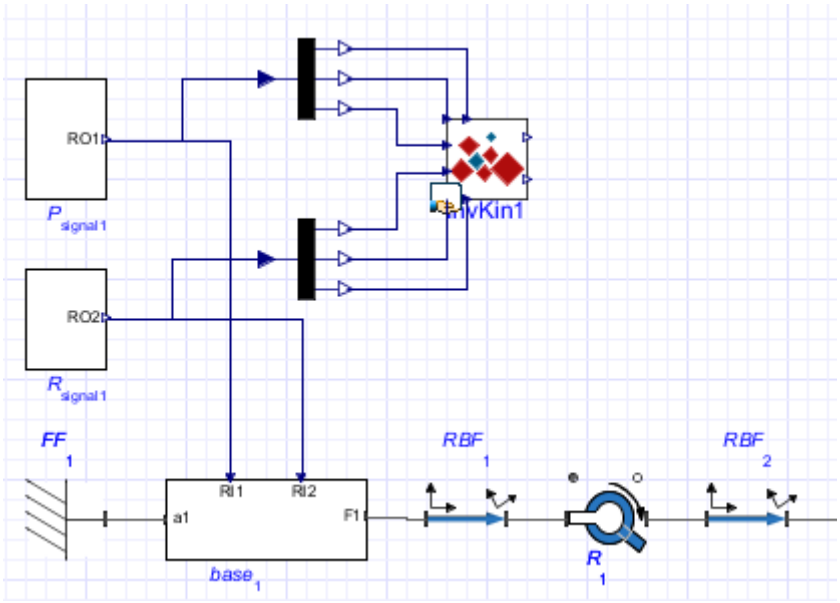


图 13-43　飞行器运动信号的连接

通过用户自定义组件 InvKin，计算 6 个飞行器运动信号的输入，得到了 2 个旋转副控制量的输出，将这两个控制量转换后输出到相应的旋转副上。对于理想模型，采用 motion driver 进行信号转换，即数字量信号是多少，相应的机械结构就运动多少，连接图如下所示。

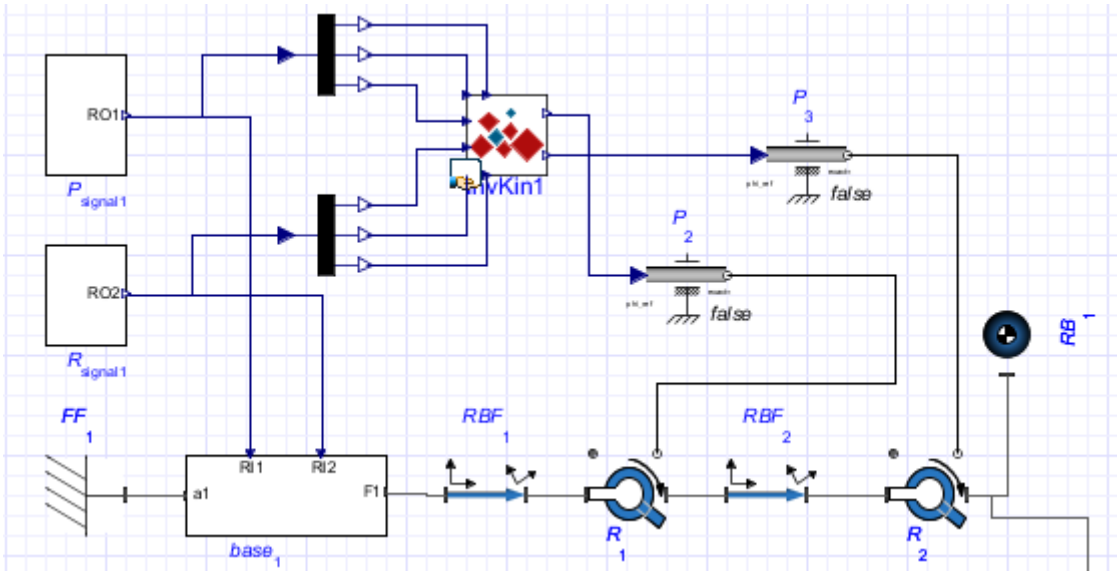


图 13-44　控制量信号的输出、转换及连接（注意信号与元件的对应）

为了验证该逆运动学计算结果的正确性，解除雷达的位置约束。即选择位置约束的滑动副、球面副 2 个元件及附近的连接线，点击右上角的  ，禁用这两个元件，取消雷达与目标点的运动约束，如下图。

图 13-45　禁用后的 2 个元件

同时添加第三段刚性框架，作为雷达射线信号的表达，如下图。因为目标点坐标为（0,0,20），所以定义第三段刚性框架矢量参数为（0,0,30）。

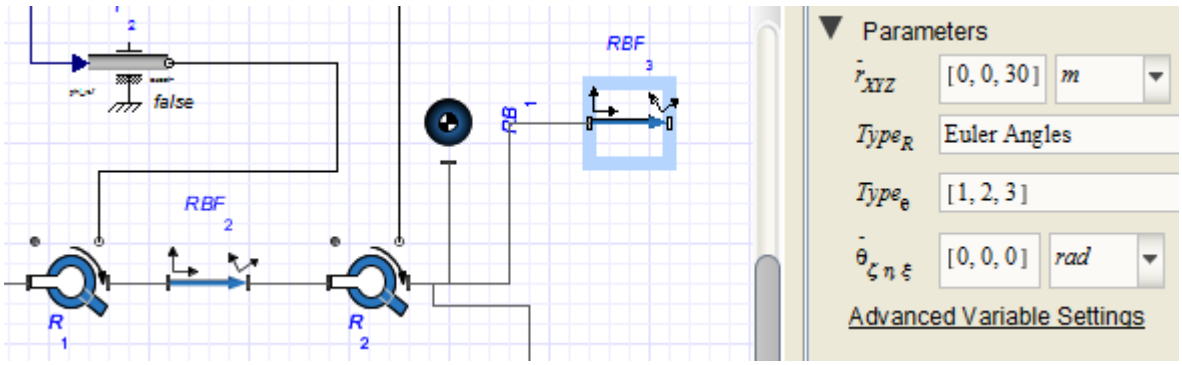


图 13-46　第三段刚性框架的参数设置

至此，该理想模型就基本完成了，点击运行  ，求解仿真结束后，可以在图形窗口观察仿真的结果，如下图。

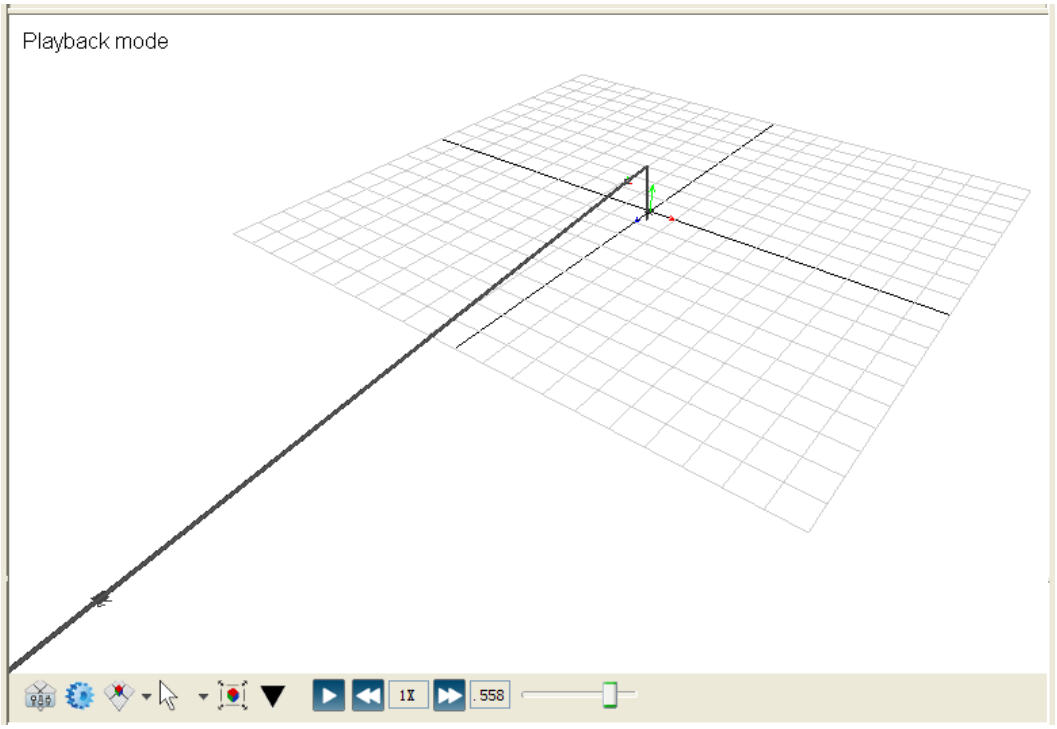


图 13-47　初步仿真结果的图形界面表达

在图形界面中，分别按住“ctrl”、“shift”、“alt”，再同时拖拽鼠标，分别可以旋转、平移、缩放模型图形，也可以通过左下角的各种设置达到相应的操作。

若对于某些零件需要得到具体的仿真结果数据（如曲线图），也可以在相应元件或连接线上添加探针 。点击该按钮，然后再点击想要考察的零件，再在右侧选择具体想要考察的物理信息，如“位置”、“速度”、“加速度”等。再重新仿真。可以得到该物理信息的具体数据曲线图，如下所示。

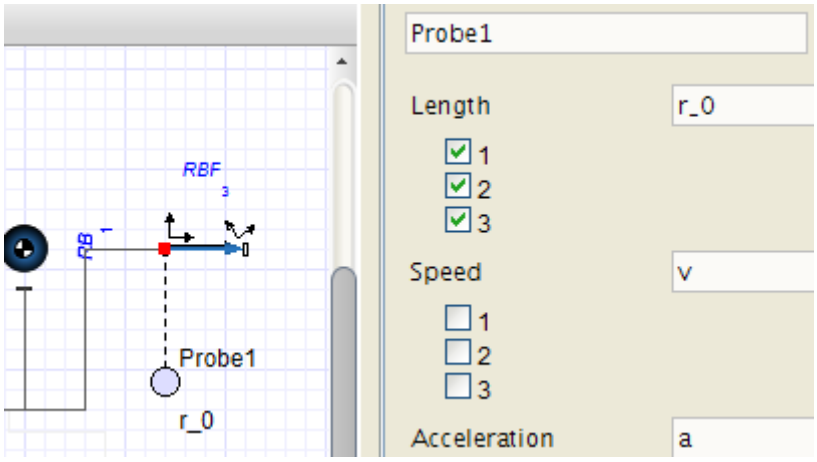


图 13-48　添加探针及选择探测的物理量

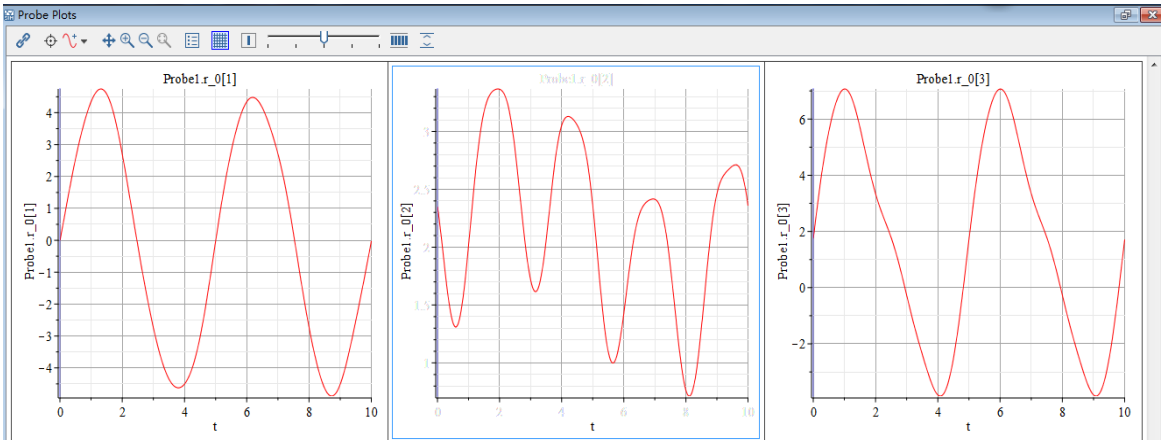


图 13-49　探针探测的物理量（第二刚性框架的 xyz3 方向的位置矢量岁时间的变化）

13.4.2 仿真后处理可视化

Maplesim 内置了许多可视化图形元件，也可以加载外部的 CAD 模型，得到更形象的仿真结果。

（1）雷达射线的可视化表达

添加可视化元件 Cylindrical Geometry（Multibody>Visualization>Cylindrical Geometry），首尾连接到第三段刚性框架上，设定其半径为 0.05m，如下图。

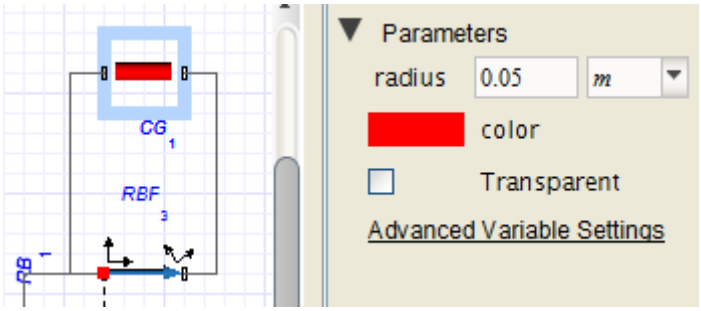


图 13-50 添加雷达射线的可视化元件和参数设置

（2）目标点的可视化表达

添加可视化元件 Spherical Geometry（Multibody>Visualization>Spherical Geometry），连接到目标点的 Fixed Frame2 上，设定期半径为 0.2m，如下图。

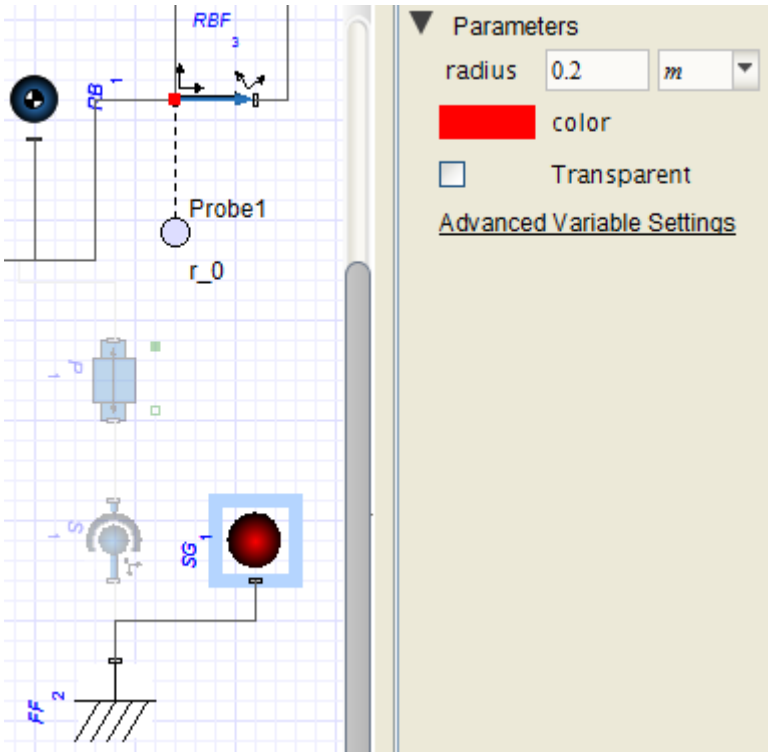
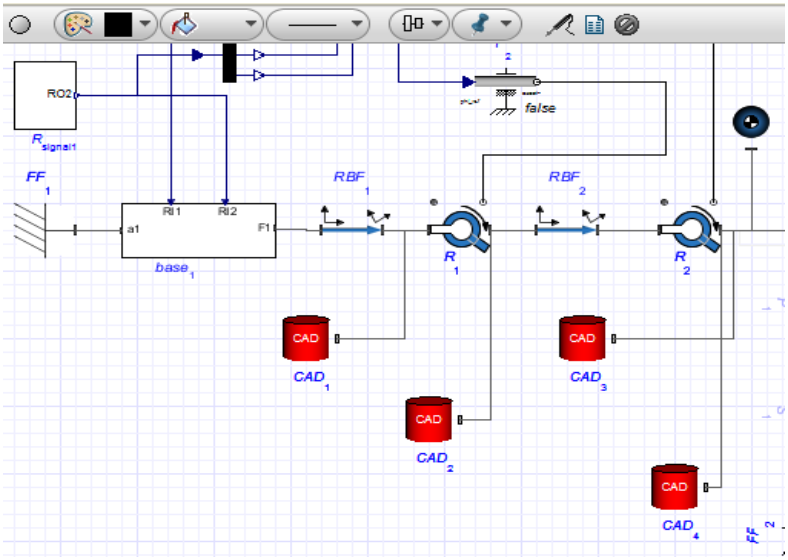


图 13-51　添加目标点的可视化元件和参数设置

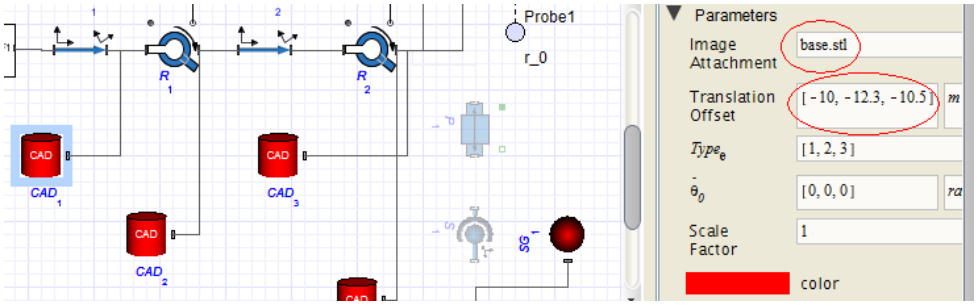
（3）各个框架的可视化表达

已经通过 CAD 建立了几个可视化模型，将它们加入模型中，与相应的元件连接起来，并设定、调整好参数。

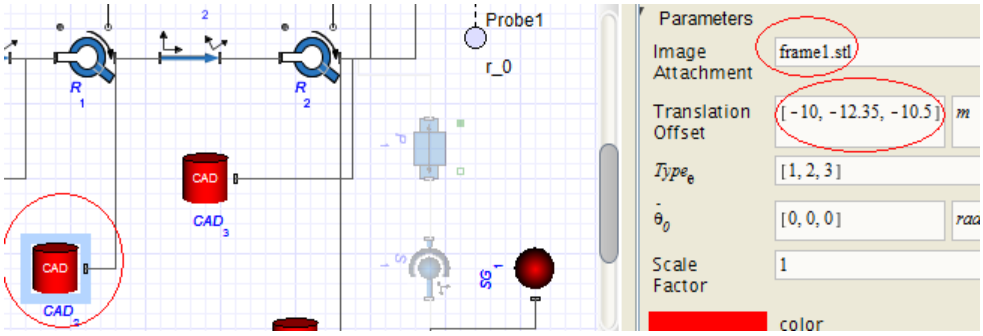
具体操作：添加四个可视化元件 CAD Geometry（Multibody>Visualization>CAD Geometry），依图进行连接并设定参数“Image Attachment”和“Translation Offset”。



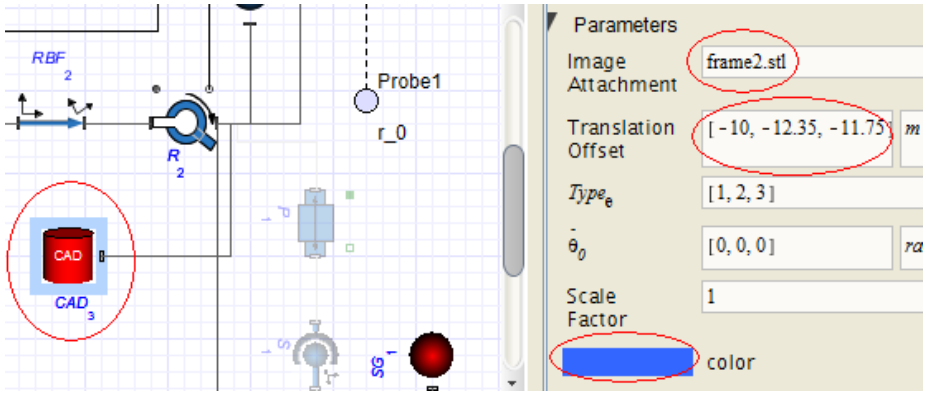
A）连接总览图



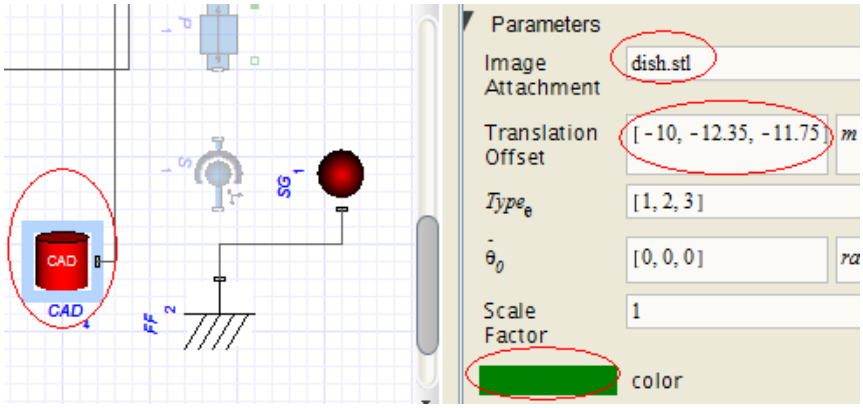
B）R_1 左侧的 CAD_1 元件参数设置



C）R_1 右侧的 CAD_2 元件参数设置



D）R_2 右侧的 CAD_3 元件参数设置



E）R_2 右侧的 CAD_4 元件参数设置

图 13-52　CAD 可视化元件的连接和参数设置

如此，该简化模型的图形化表达也建立完成了，再次运行仿真，结果如下。

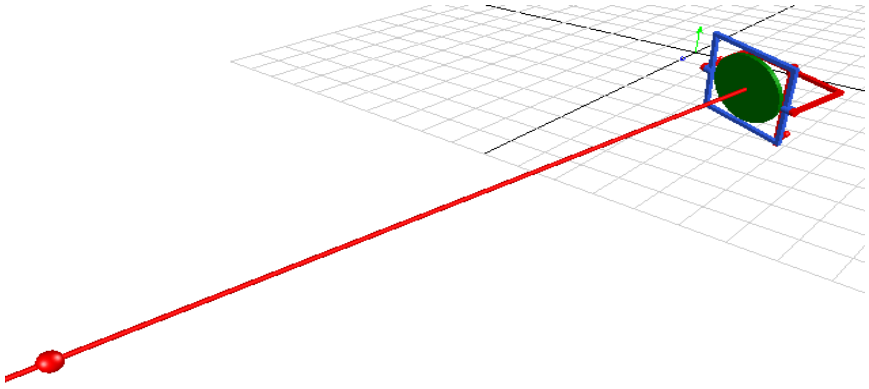


图 13-53　添加可视化图形后的仿真结果

在动画演示中，许多元件的内含图形也会表达出来，点击隐藏元件的默认图形，就可以得到完全由可视化图形表达的仿真结果了（上图已隐藏）。

13.4.3 PID 控制系统

以上模型的控制系统，使用的是理想的开环控制。许多情况下，面对实际问题需要对系统进行闭环控制，以针对非理想情况而提高控制精度，我们也可以在 maplesim 中添加各种控制方式。下面提供该例的 2 种 PID 控制方案，仅供参考。

13.4.3.1 使用 PID 元件的闭环控制

具体操作：如下图，删除 2 个 Rotational Position 元件及其连接线。

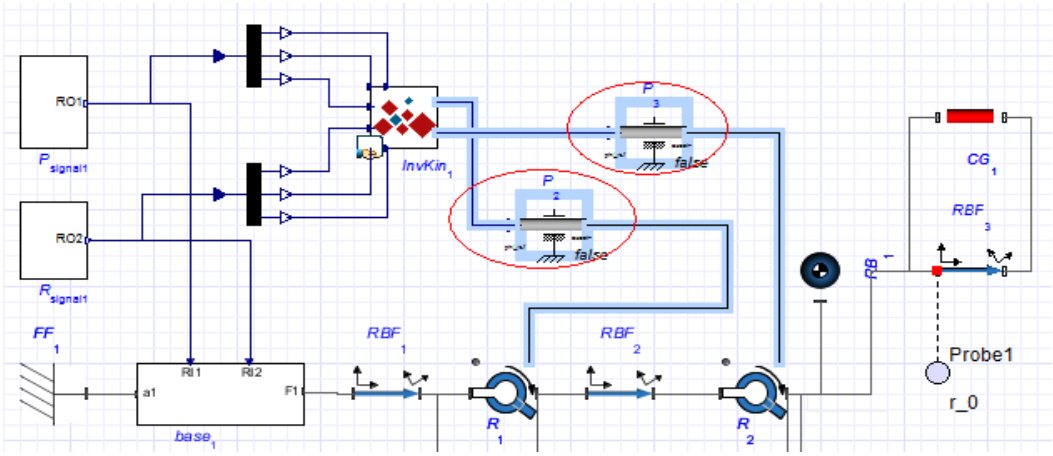


图 13-54　删除 2 个 Rotational Position 元件及其连接线

在搜索栏中输入“PID”，选择“PID”，然后拖入旁边的 PID 元件图标。该元件也可以通过 Libraries>Signal Blocks>Common>PID 选择。选择新加入的 PID 元件，建立新的子系统（ctrl+g），命名为 R_PID，如下图。

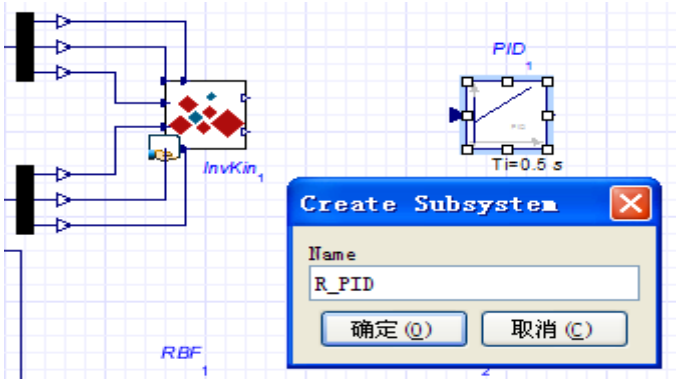


图 13-55　建立新的子系统 R_PID

进入该子系统，按照一般反馈调节的系统结构，添加、连接元件，如下图所示。

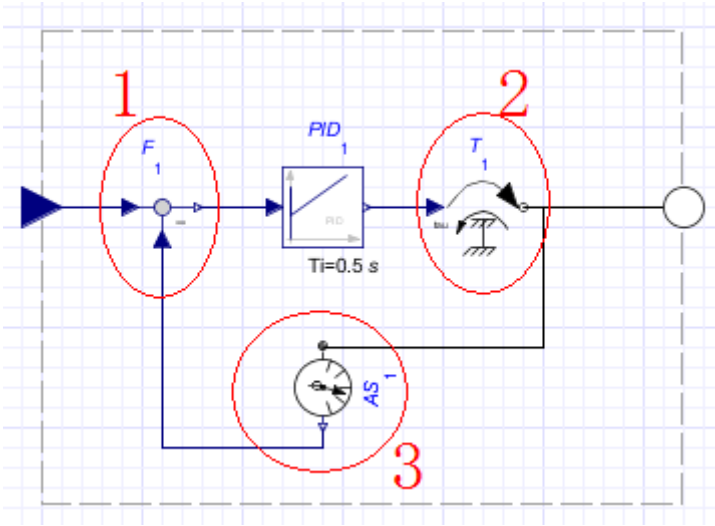


图 13-56　编辑新的子系统，添加 3 个元件

上图中各个元件路径如下：（也可通过搜索查找）

元件 1：Signal Blocks>Common>Feedback

元件 2：1-D Mechanical>Rotational>Torque Drivers>Torque

元件 3：1-D Mechanical>Rotational>Sensors>Angle Sensor

返回主模型，选择该新子系统，右键，选择 Convert to Shared Subsystem，将该子系统共享，点击确定，如下图。

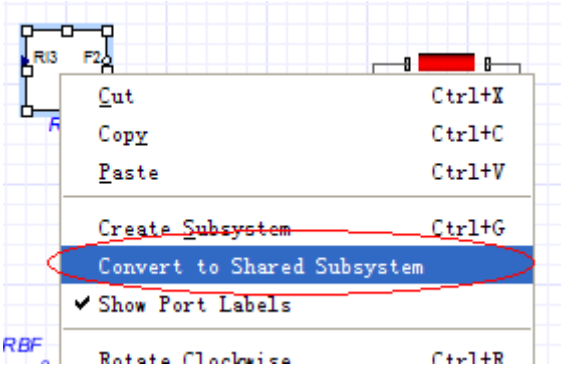


图 13-57　将新的子系统共享

如此，在左侧的 Project>Definitions>Subsystem 中会得到刚刚共享的新子系统 R_PID，如下图。可以在其他需要的地方调用该子系统，并且对任意一个该子系统进行修改时，其他该子系统也会发现同样的修改。

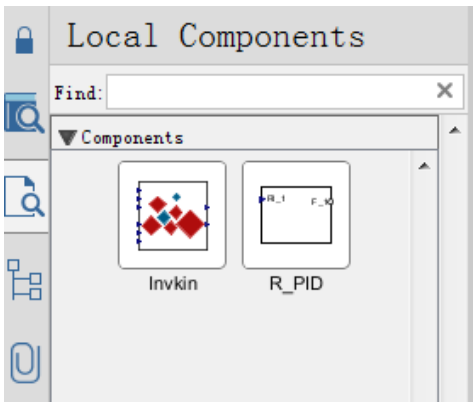


图 13-58　共享后的子系统

再添加一个新子系统 R_PID，然后如下图进行连接。

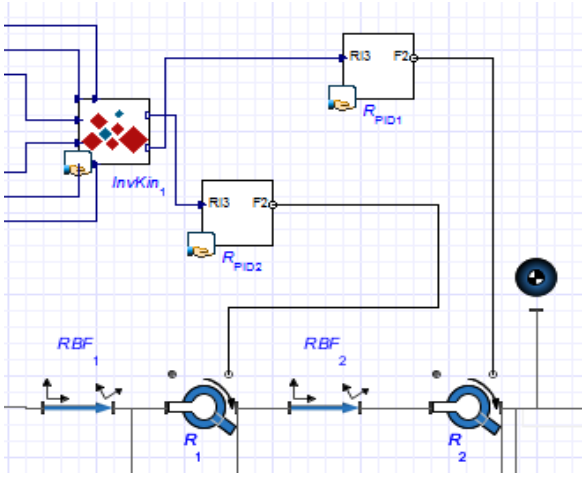


图 13-59　加入第二个 R_PID 并依图连接

如此就建立了一个局部的 PID 控制。

PID 控制器的三个参数的需要整定，可以建立一些探针，然后用类似实验的方法进行整定；也可以使用 maplesim 的 ControlDesignTools 工具包进行参数整定。

在此，直接给出一组较理想的 PID 参数，如下图（P=1000，I=10，D=5）

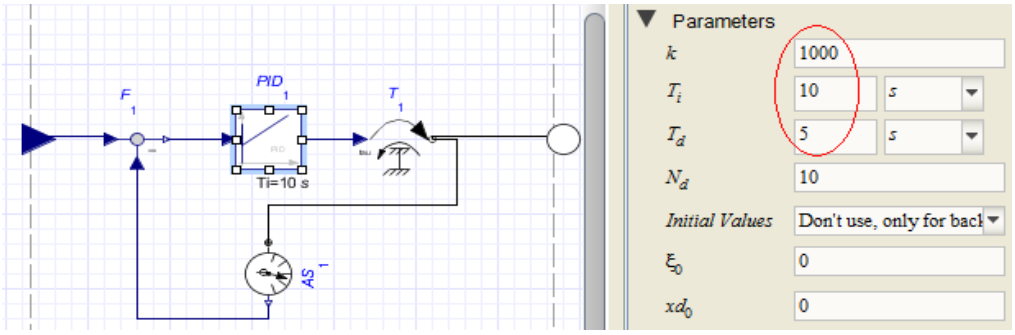


图 13-60　PID 控制器的 P、I、D 参数设定

点击运行，观察仿真结果，如下图。

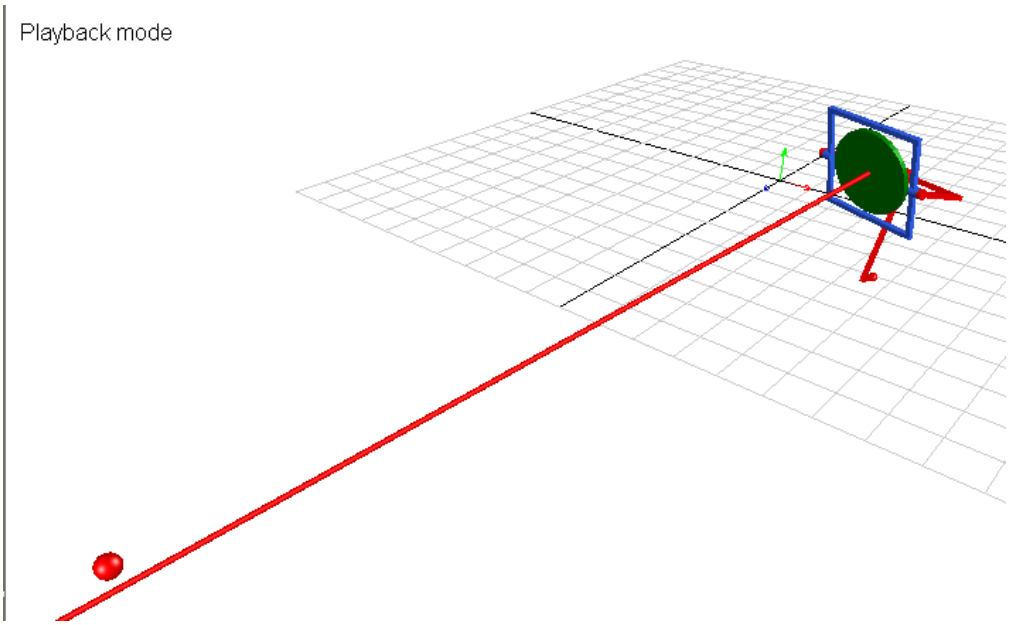


图 13-61　添加 PID 控制器的仿真结果

13.4.3.2 使用 Lim PID 元件的闭环控制

Lim PID 元件的结构与上例中使用的 PID 元件结构有所不同，具体可以分别点击进入 2 个元件观察内部结构，下面使用 Lim PID 元件建立类似上例的闭环控制。

具体操作：在搜索栏中搜索“pid”，选择“Lim PID”。在 R_PID 子系统中删除原有

所有元件，拖入 Lim PID 元件，然后继续如图添加下列元件，然后同时设置 PID 的参数为 P，I，D 及其他参数，如图。

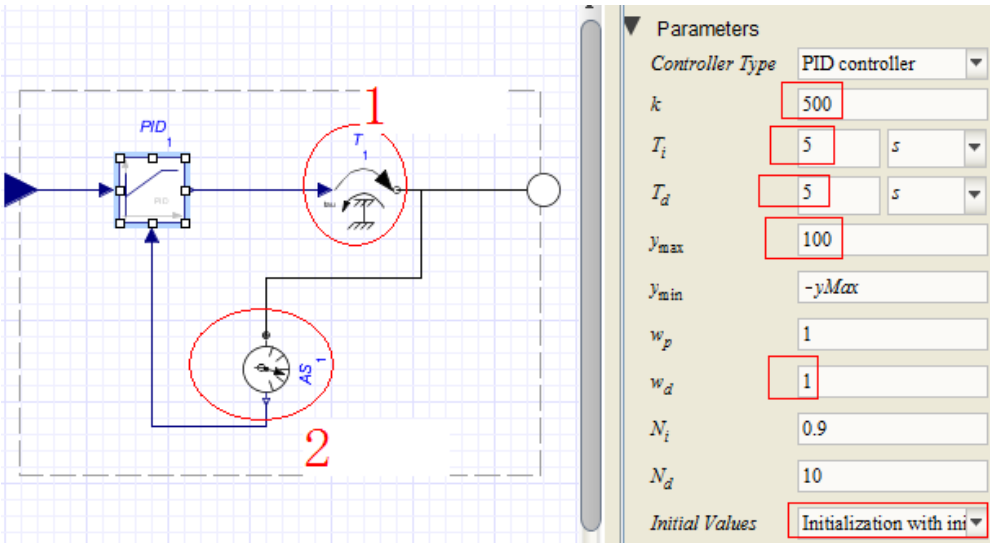


图 13-62　新的 R_PID 子系统构成及 Lim PID 元件参数设置

元件 1：Libraries>1-D Mechanical>Rotational>Torque Drivers>Torque

元件 2：Libraries>1-D Mechanical>Rotational>Sensors>Angle Sensor

再次仿真，结果如下：

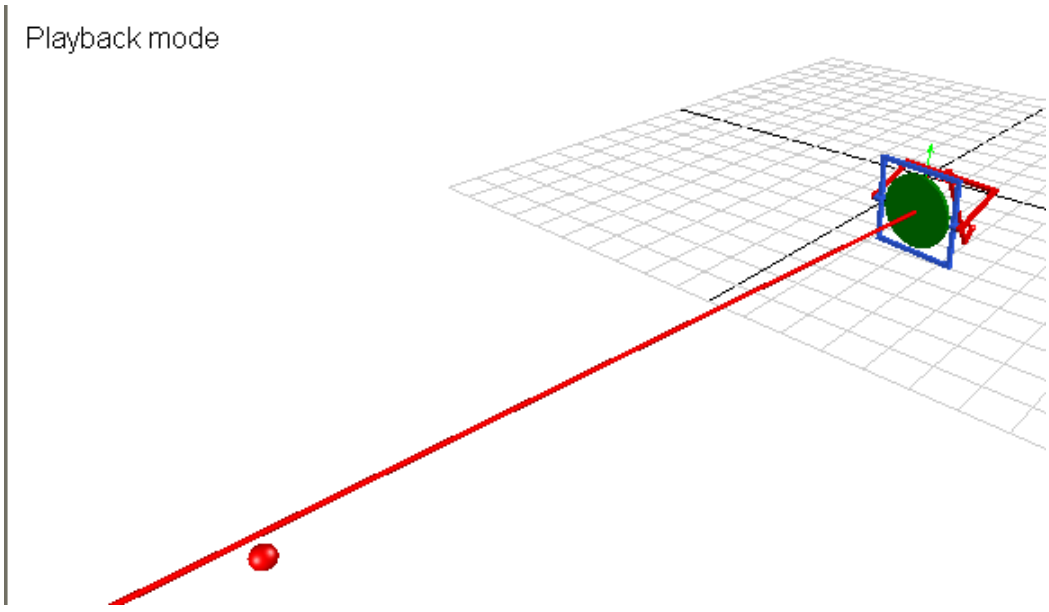


图 13-63　新的 PID 控制的仿真结果

第十四章 接触模型与蒙特卡洛分析

14.1 接触模型与蒙特卡洛分析

模型下载地址：http://www.maplesoft.com.cn/book/14.1.zip

1、设置重力方向为 Z 轴负方向。

建立下落的球体模型，如下图：

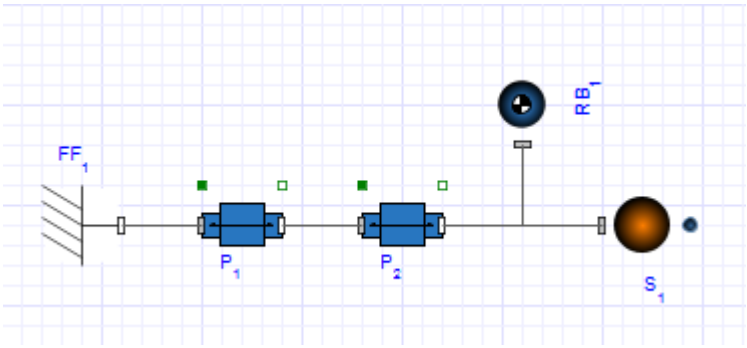


图 14-1 下落球体模型

元件位置分别为：

Multibody>Bodies and Frames>Fixed Frame

Multibody>Bodies and Frames>Rigid Body

Multibody>Joints and Motions>Prismatic

Multibody>Contacts>Elements>Sphere

2、 参数设置分别如下：

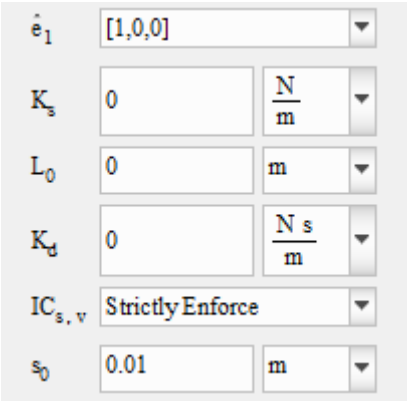


图 14-2 设置 P_1 参数

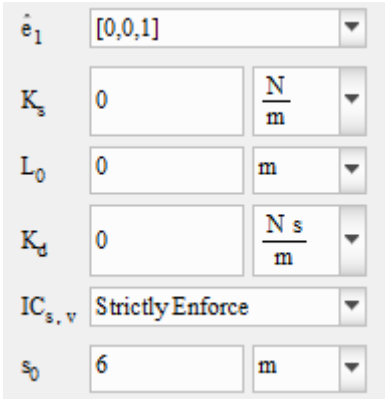


图 14-3　设置 P_2 参数

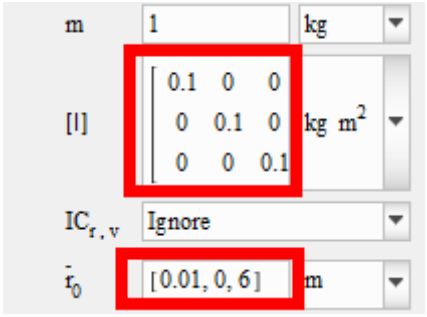


图 14-4　设置 RB_1 参数

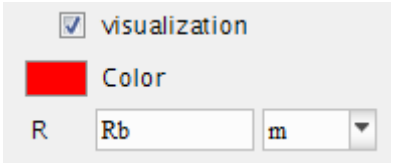


图 14-5　设置 S_1 参数

3、将其打包为一个子系统 ball

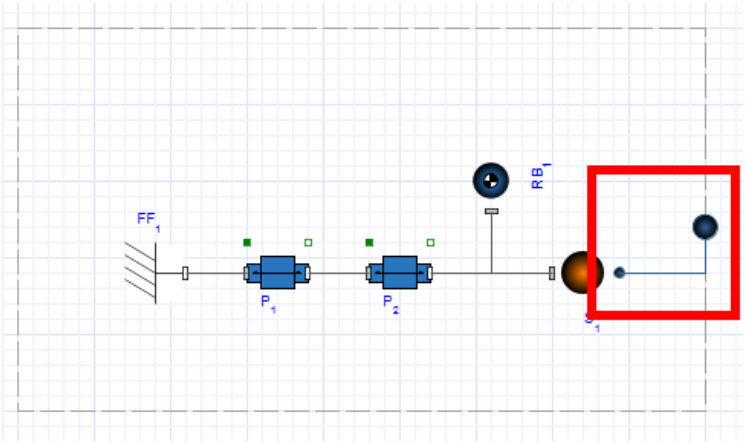


图 14-6　创建子系统 ball 并建立端口

4、创建一个固定的碰撞球模型

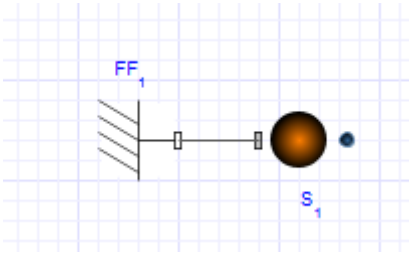


图 14-7　固定的碰撞球模型

元件位置：

Multibody>Bodies and Frames>Fixed Frame

Multibody>Contacts>Elements>Sphere

5、参数设置

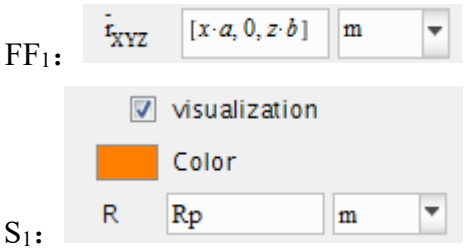
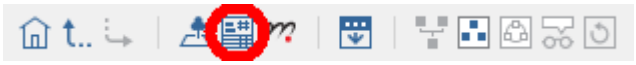


图 14-8　定义碰撞球模型中的参数

子系统参数：



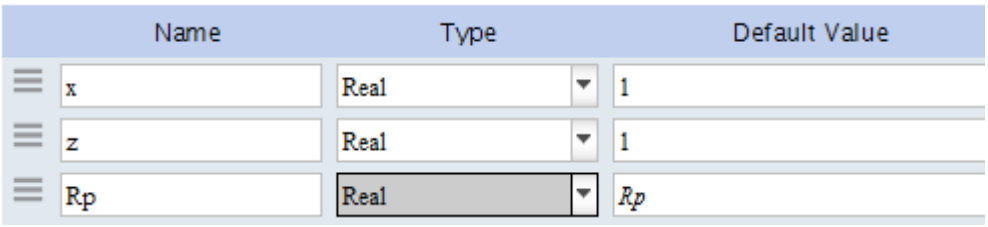


图 14-9　定义子系统的参数

6、将其打包成一个共享的子系统 pin

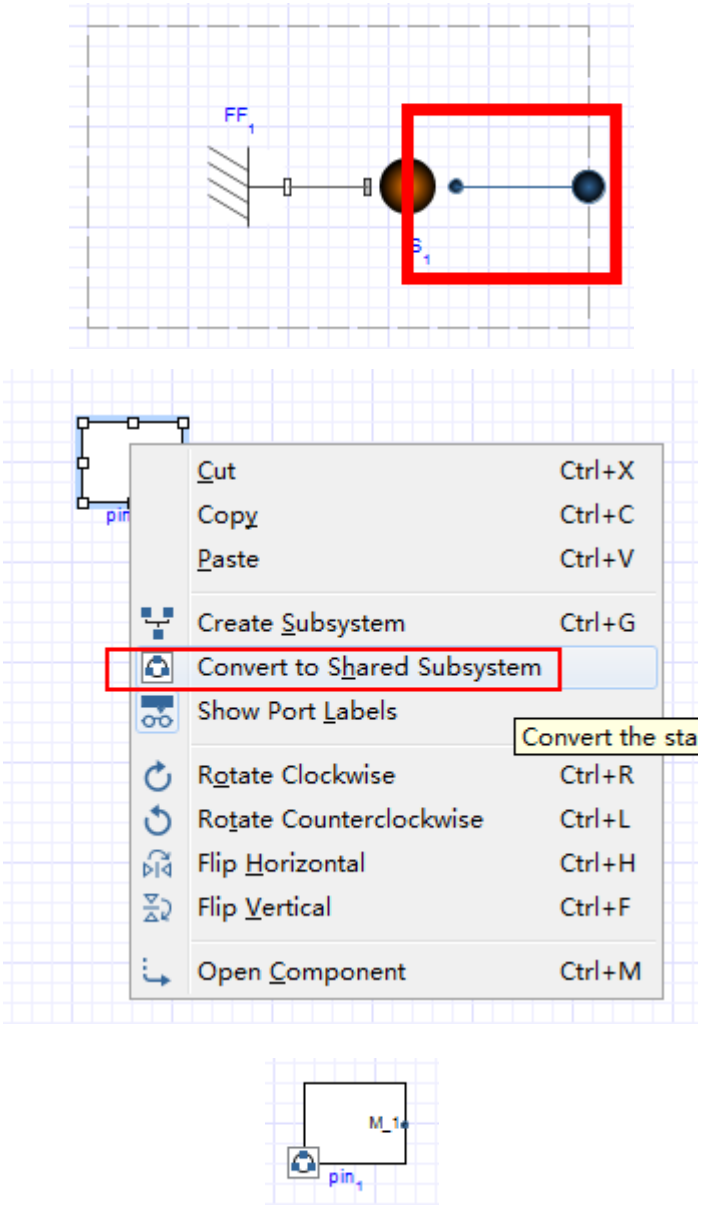


图 14-10　固定的碰撞球模型建立的子系统

7、克隆多个固定的碰撞球

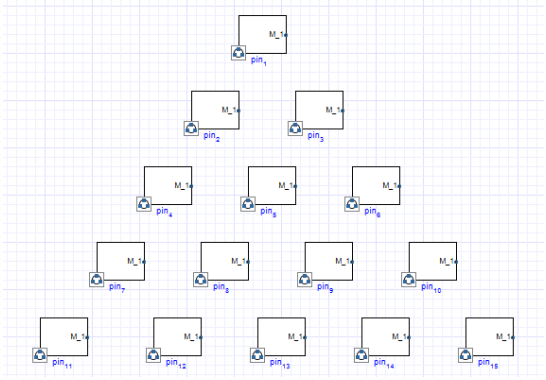


图 14-11　复制多个子系统

8、创建地面的接触模型

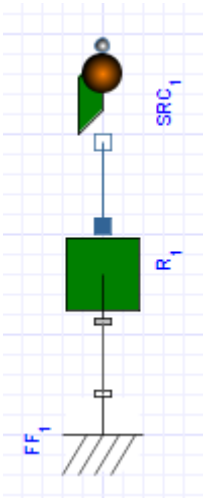


图 14-12　地面接触模型

元件位置：

Multibody>Bodies and Frames>Fixed Frame

Multibody>Contacts>Elements>Rectangle

Multibody>Contacts>Forces>Sphere Rectangle Contact

9、参数设置

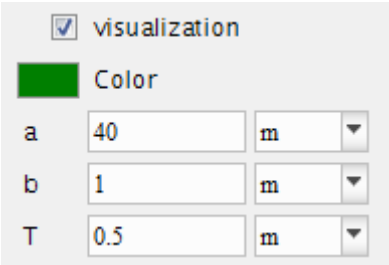


图 14-13　设置 R_1 的参数

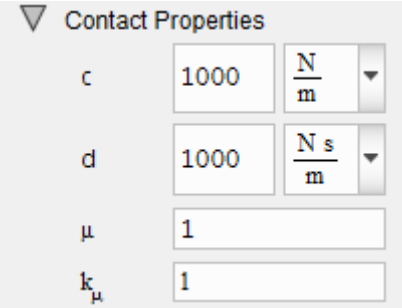


图 14-14　设置 SRC_1 的参数

10、连接各个多体间的接触

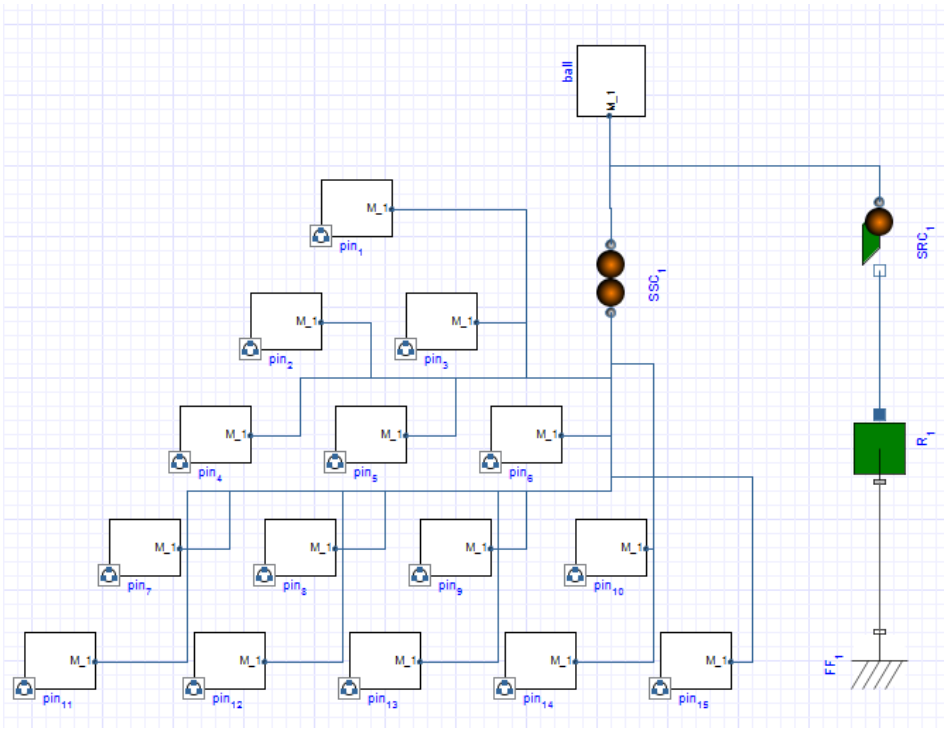


图 14-15　各个元件之间的连接

Multibody>Contacts>Forces>Sphere Sphere Contact

注意各个固定碰撞球与下落球的连接：

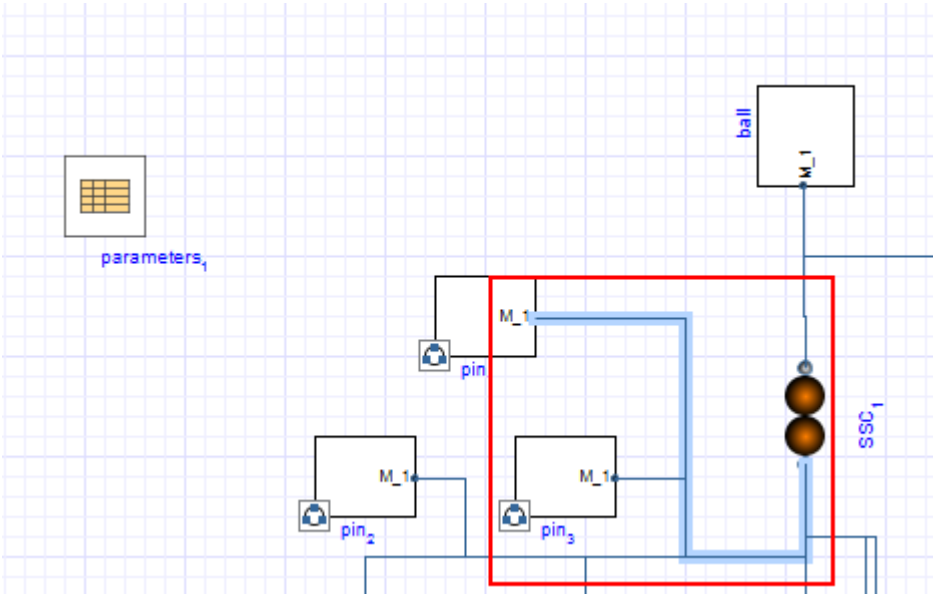


图 14-16　pin 子系统与 SSC_1 之间的连接

11、参数设置

各个 pin 的坐标设置，按下图进行调整，具体请参考模型：

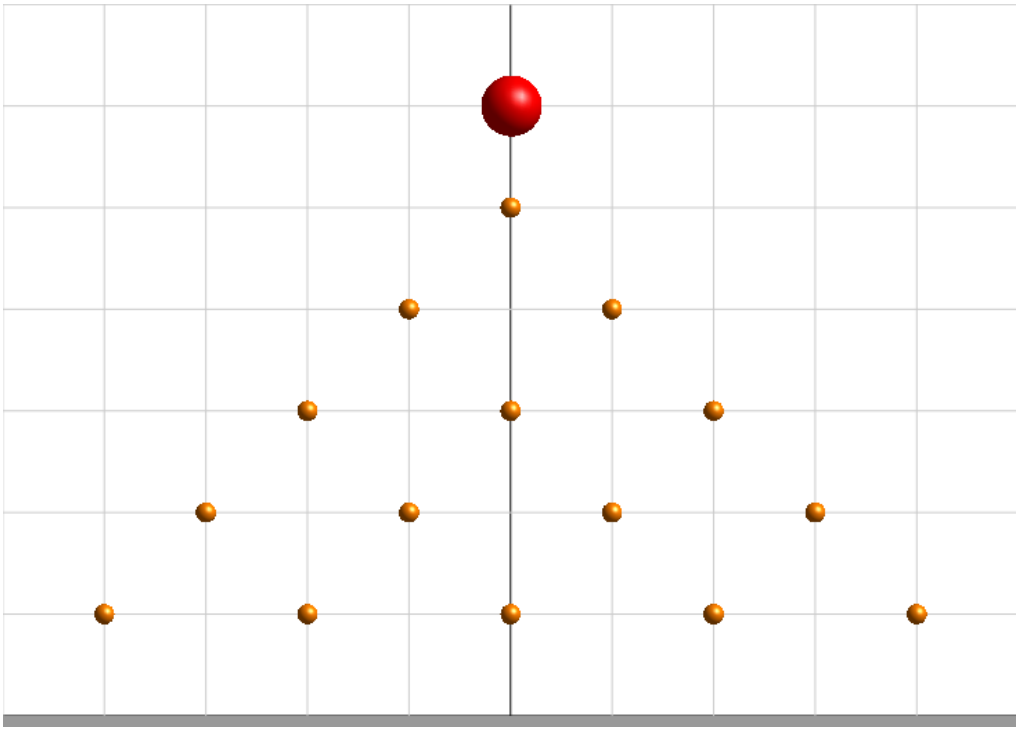


图 14-17　3D 模型图

下落球与固定碰撞球接触参数：

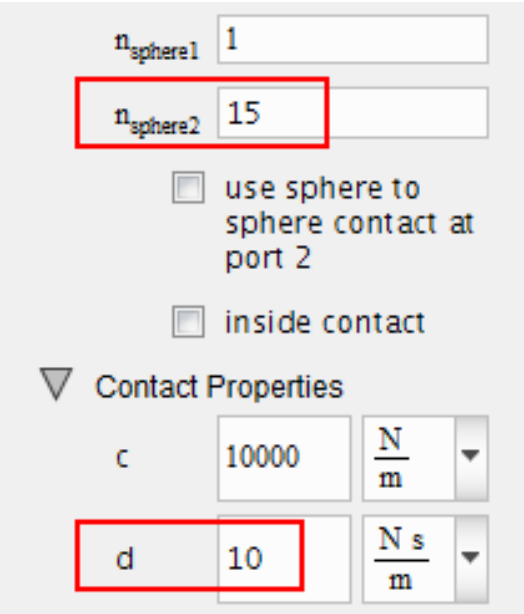


图 14-18　设置 SSC 的参数

12、创建参数表

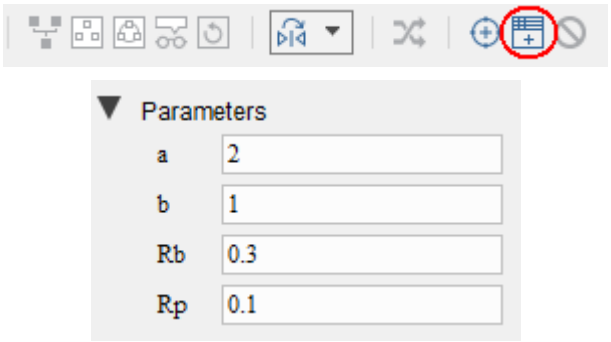


图 14-19　创建参数表元件

13、设置探针，仿真

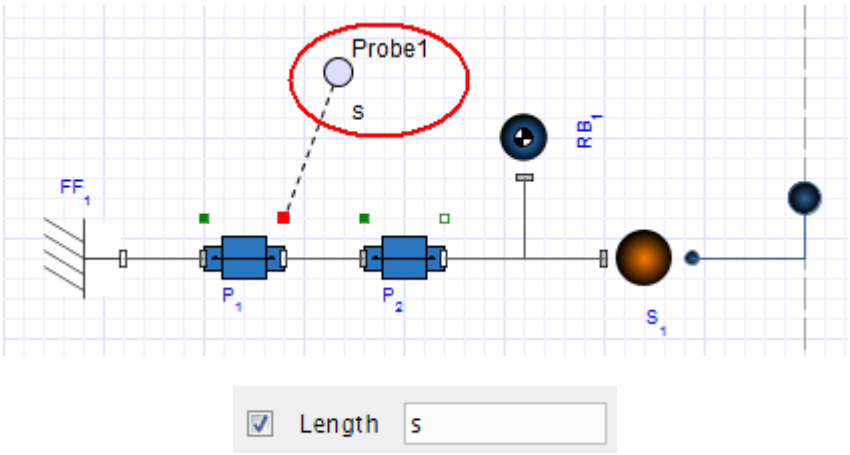


图 14-20　添加探针并仿真

第十五章 MapleSim 模型自动代码生成以及与 VC 项目的集成

15.1 介绍

本节内容将介绍如何利用 MapleSim 中的 Code Generation 模板生成模型的 C 代码，然后在 Visual C++ 2015 Express 中编译这些代码。MapleSim 可以生成任意模型的 ANSI C 代码，代码包含描述模型动态行为的微分方程以及求解器。生成的 C 代码是免费的，可以脱离 MapleSim 环境在其他任意的仿真工具或开发项目中使用。

我们将通过案例说明如何生成一个可执行项目，并输出仿真结果到数据文件中。您可以使用相同的方法在其他开发项目中使用 MapleSim 中的 code generation。

注意：MapleSim 生成的代码中包含求解器，用户仅需要编写 main()函数。Main()函数的详细说明见 Maple 中的帮助 GetCompiledProc。

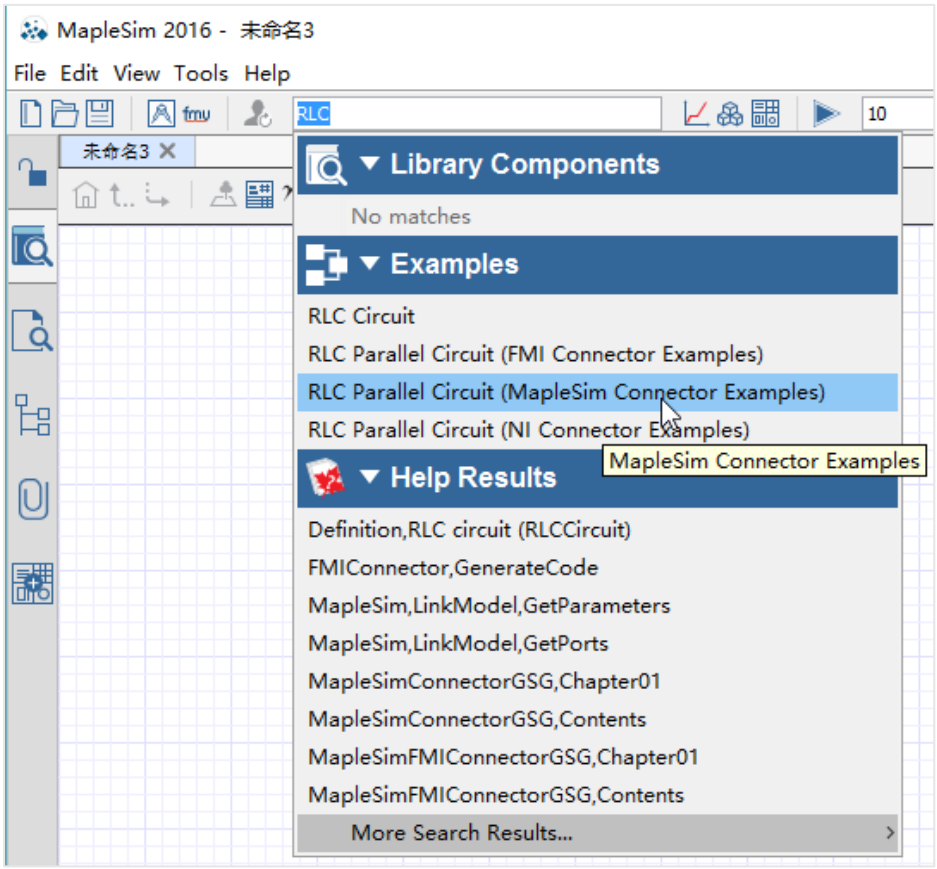
模型和代码下载地址：http://www.maplesoft.com.cn/book/15.1.zip

15.2 使用的工具

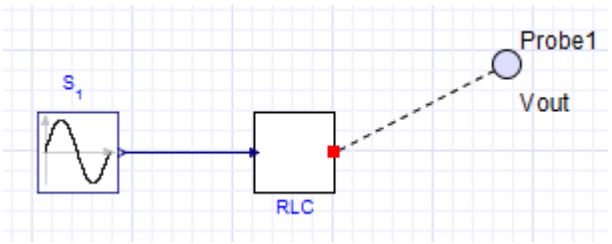
- 64 位 Windows 7 或 10
- MapleSim 2016.2
- Microsoft Visual Studio Express 2015 for Windows Desktop
  详细信息见 Maple 帮助文件：compiler,setup
- Excel（版本不限）

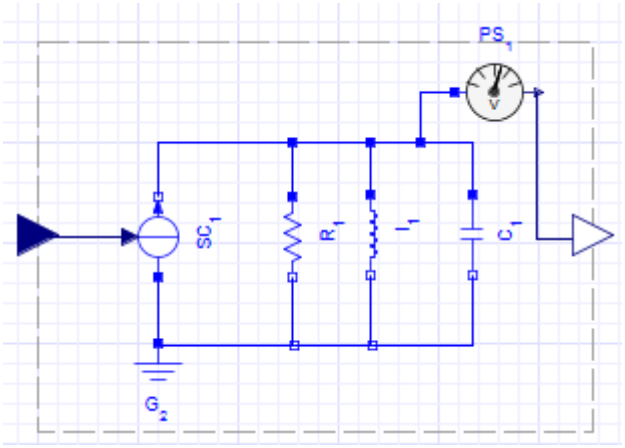
15.3 准备 MapleSim 模型

首先，我们在 MapleSim 创建一个模型。打开 MapleSim 软件，在搜索区域输入 RLC，在弹出的模型示例列表中选择 RLCParallel Circuit (MapleSim Connector Example)，在 MapleSim 打开该模型。



这里我们使用了一个 RLC 电路模型，它包含一个子系统（我们需要生成这部分的 C 代码），有一个实值输入和一个实值输出。

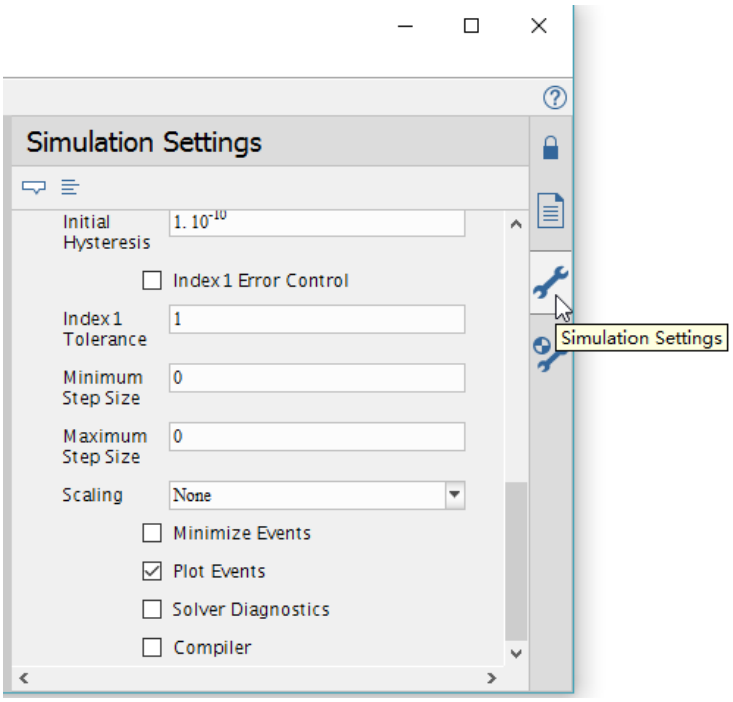




注意：C 代码输出的结果仅是子系统的输出，并不输出 probe 的结果。

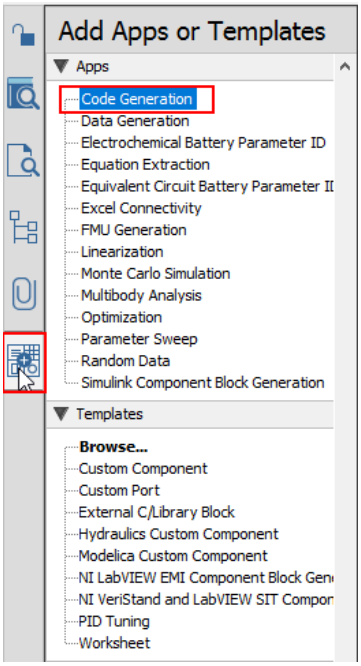
注意：以下步骤是 MapleSim 2016 中的一个 bug，在其他版本中可以不进行如下操作。

展开 Simulation Setting 面板，取消 Plot Events 参数项。

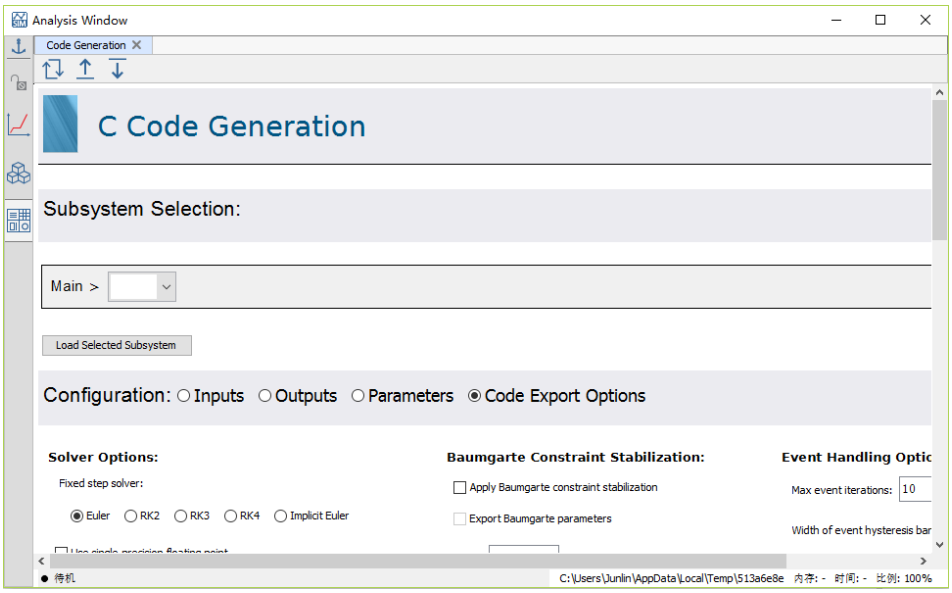


15.4 使用 MapleSim 内置的 Code Generation 模板

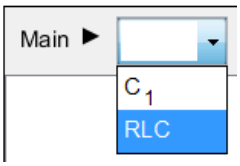
1) 点击左侧的面板 Add Apps or Templates > Apps > Code Generation

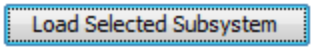


2) 双击或右键点击 Code Generation，打开 Code Generation Template：

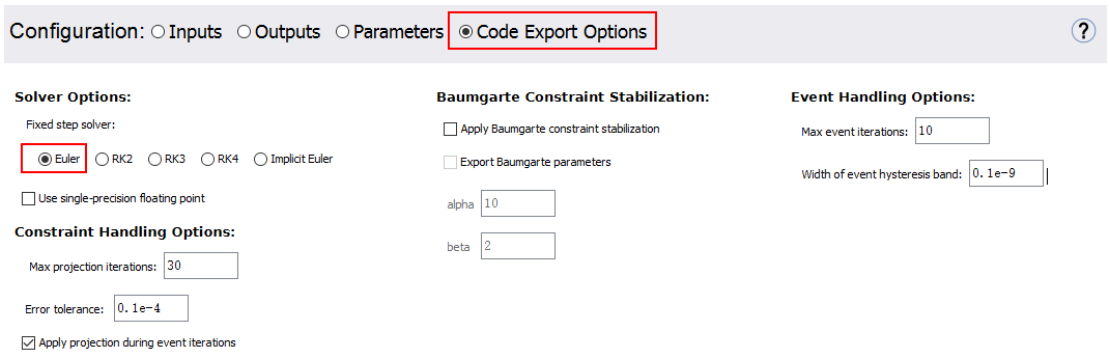


3) 选择子系统 RLC：

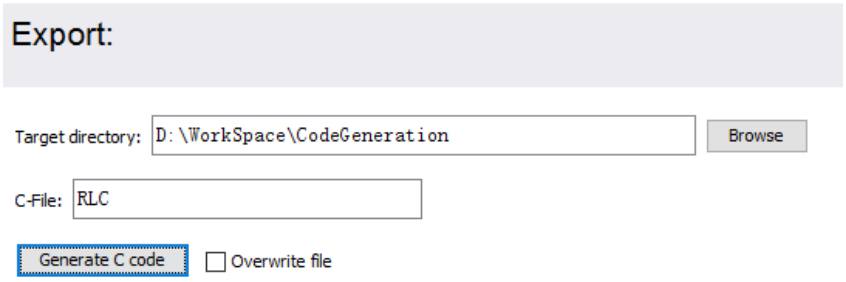


4) 点击 Load Selected Subsystem 

5) 进入 Configuration 区域，设置参数。选择 Euler 求解器：



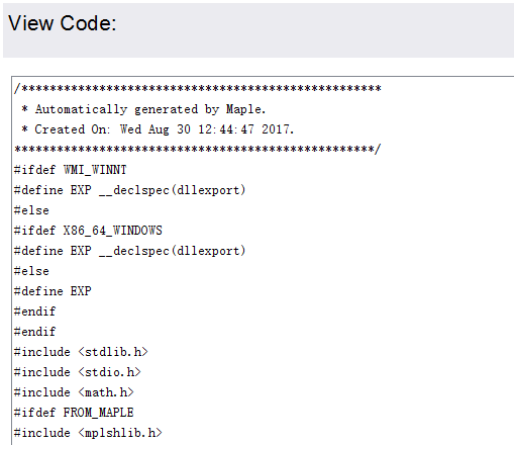
6) 进入 Output 区域。生成并保存 C 代码，选择保存路径和文件名：



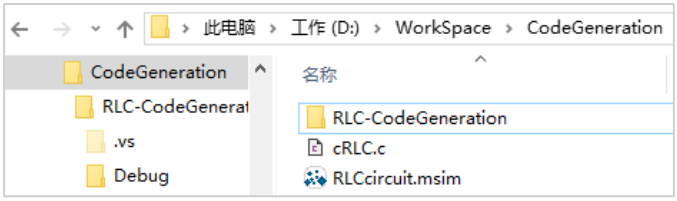
MapleSim 将自动在文件名前加上前缀名 c，文件格式为 c。例如上图中的设置将生成 cRLC.c 文件。

点击下面的 Generate C Code 

7) 进入 View Code 区域。C 代码将显示在下面的文本区域



同时也会保存到指定的路径。



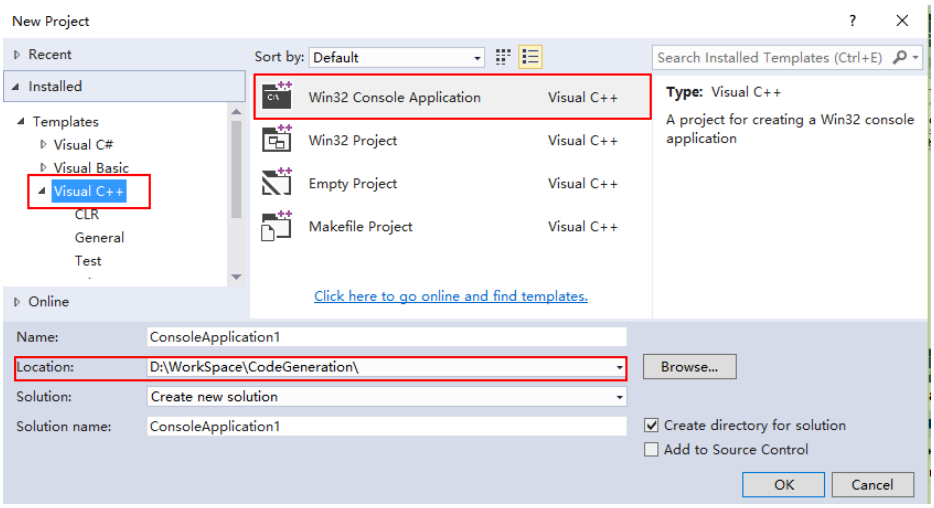
注意：仿真的输入是由代码中的 inpfn() 给出。

```
static void inpfn(double T, double *U)
{
 U[0] = sin(T);
}
```

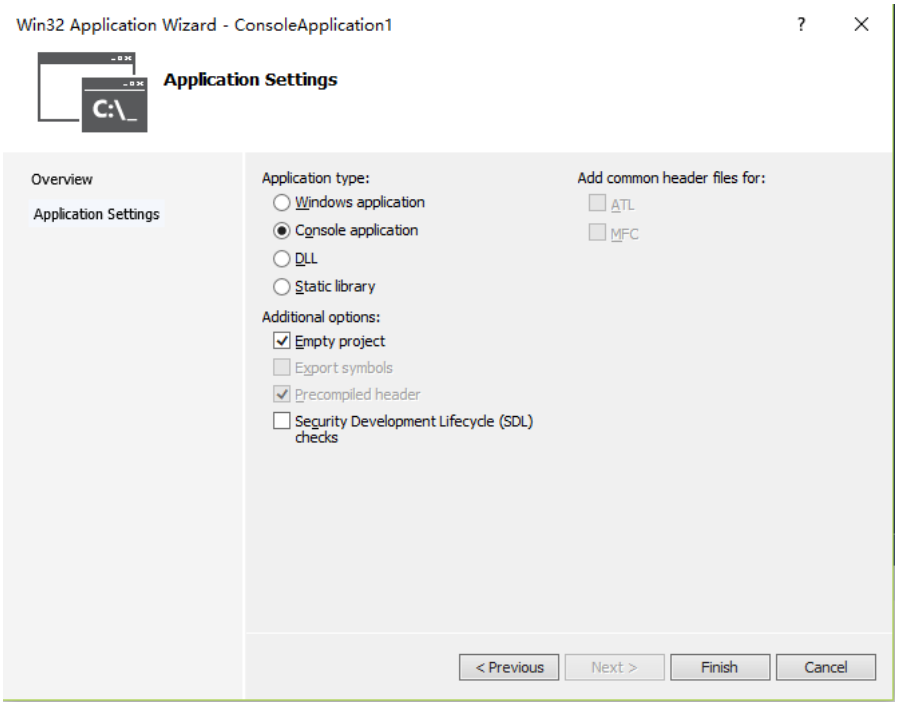
默认的输入时正弦曲线，但是你可以根据需要修改。如果 MapleSim 有两个或更多的输入，将通过代码文件中的 U[0], U[1]…U[n]指定。

15.5 在 Visual C++ Express 2015 中创建一个新的项目

在 Visual C++ 2015 Express 中，找到文件 File > New Project，选择 Win32 Console Application。指定项目名和保存位置，然后点击 OK > Next。



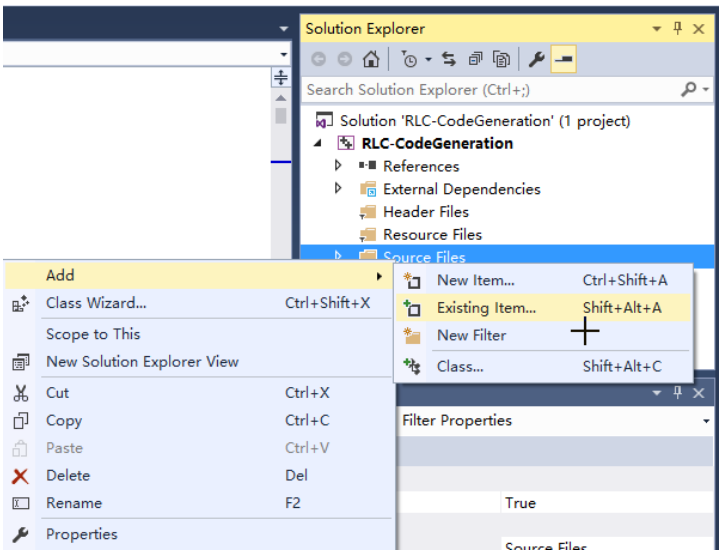
在 Application Setting 窗口中，选中 Empty project，确保 Precompiled header 没有被选中。



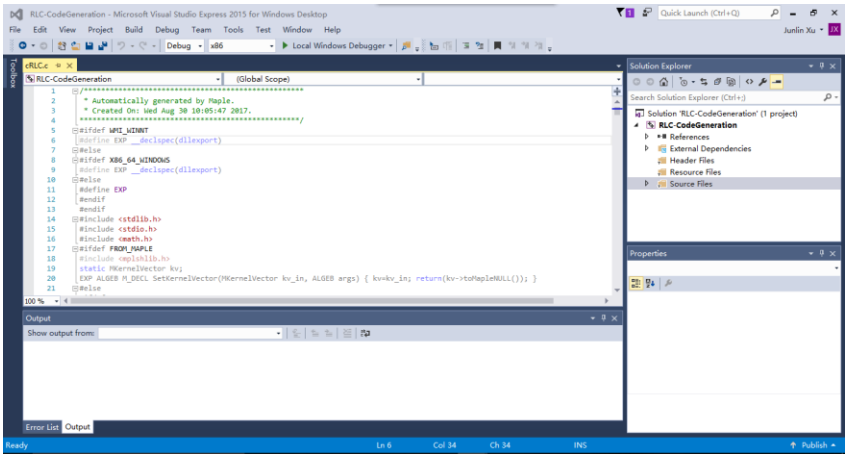
点击完成按钮。

15.5.1 添加 MapleSim 生成的 C 代码到项目中

在 Solution Explorer 中，右键点击 Source Files，然后选择 Add > Existing item…



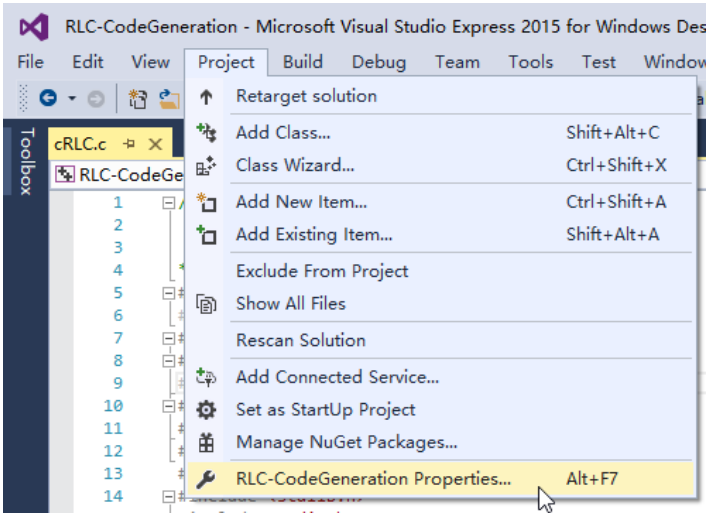
浏览和选择 MapleSim 生成的 C 代码文件 cRLC.c，现在你已经在 Visual Studio 项目中加入了 C 代码。

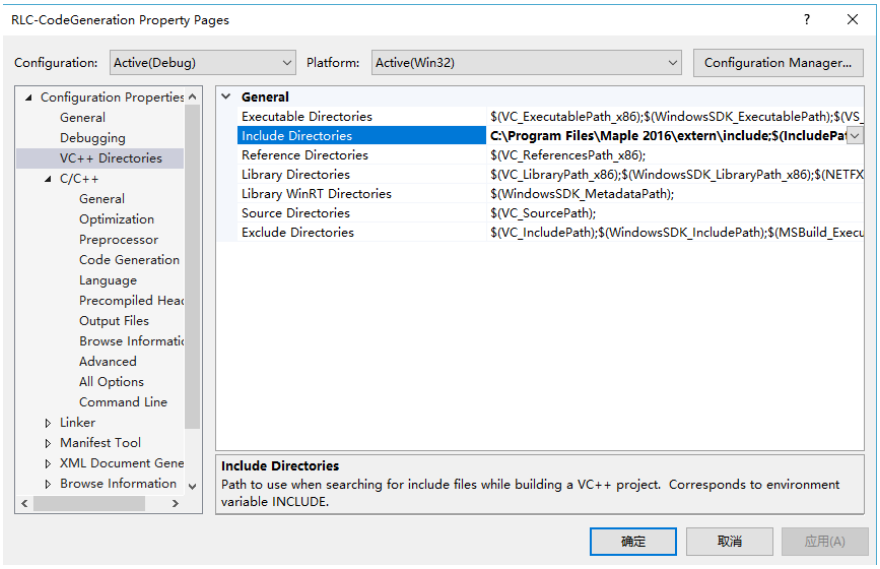


15.5.2 定义 Include 文件和编译器位置

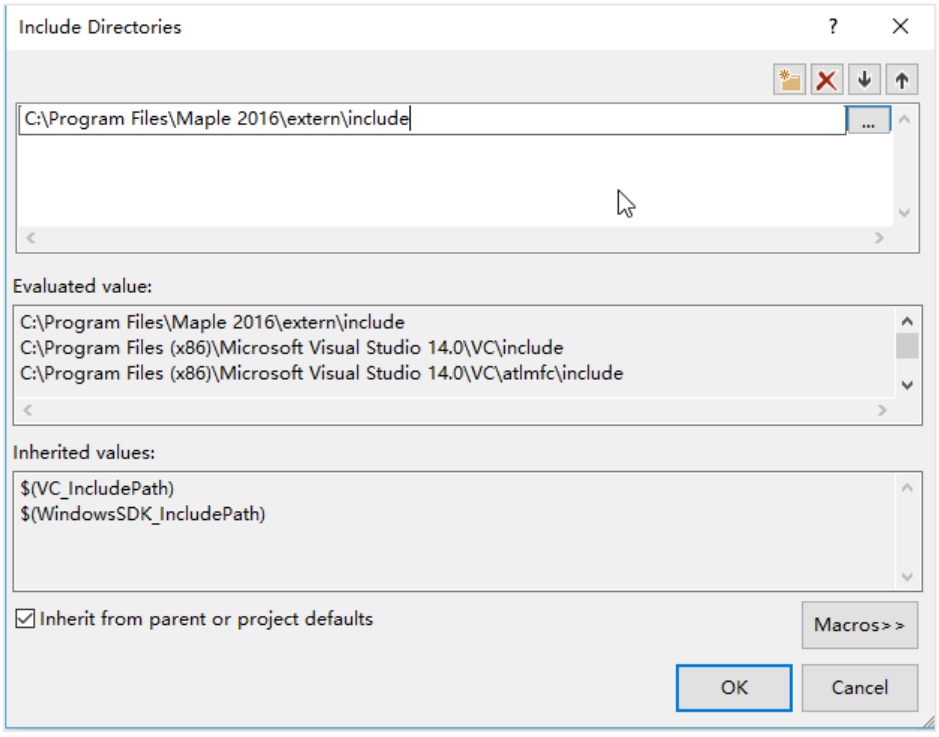
定义 Include 文件和编译器设置：

找到菜单 Project > Properties，选择 VC++ Directories，在右侧的面板中点击 Include Directories。

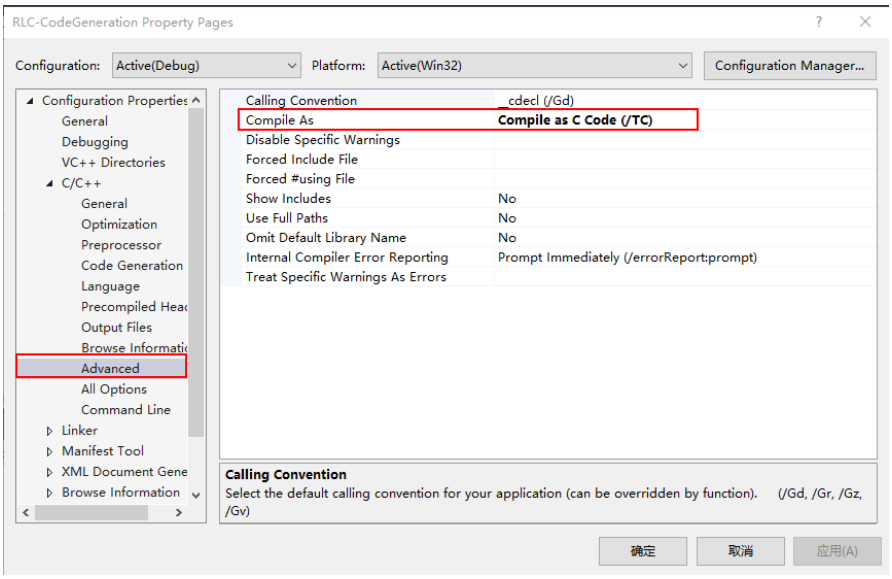




添加 C:\Program Files\Maple 2016\extern\include 到路径。你的 Include Directories 路径将显示如下：



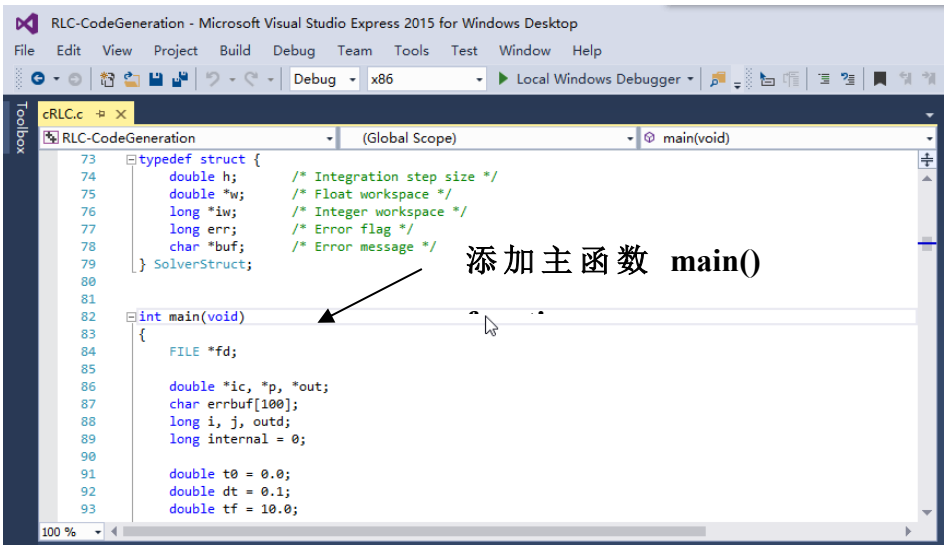
找到 Configuration Properties > C/C++ > Advanced。在右侧的面板中，改变 Compile As 设置为 Compile As C Code (/TC)。



点击确定按钮。

15.5.3 添加 main()文件到代码中

返回到 Visual Studio 中的源代码。



添加下面的 main() 函数到代码中（例如 cRLC.C 文件）。

```
int main(void)
{
    FILE *fd;

    double *ic, *p, *out;
    char errbuf[100];
    long i, j, outd;
```

```
    long internal = 0;

    double t0 = 0.0;
    double dt = 0.1;
    double tf = 10.0;

    long npts = (long)ceil((tf + 1e-10 - t0) / dt) + 1;

    out = (double *)malloc(((npts)*(NOUT + 1) + NOUT + 1) * sizeof(double));

    fd = fopen("RLC-Output.dat", "w");

    outd = ParamDriverC(t0, dt, npts, NULL, NULL, out, &errbuf, internal);

    for (i = 0; i<npts; i++)
    {
        fprintf(fd, "%lf ", out[i*(NOUT + 1)]);
        for (j = 0; j<NOUT; j++)
        {
            fprintf(fd, "%lf ", out[i*(NOUT + 1) + j + 1]);
        }
        fprintf(fd, "\n");
    }
    fclose(fd);

    return 0;
   }
```

通过 t0, dt, tf 分别定义仿真开始时间，时间步，结束时间。ParamDriverC() 的调用包含了状态变量，通过 inpfn() 定义的输出。

修改输入信号，如下：

```
static void inpfn(double T, double *U)
{
    U[0] = 1.;
}
```

关于接口函数中的参数说明，详细信息请参阅?GetCompiledProc

仿真结果将写入到 output.dat 文件中。

15.5.4 C 代码说明

MapleSim 生成的 C 代码包含 4 个重要的函数：

- SolverSetup(t0, *ic, *u, *p, *y, h, *S)
- SolverStep(*u, *S)

  这里 SolverStep 是 EulerStep, RK2Step, RK3Step 或者 RK4Step
- SolverUpdate(*u, *p, first, internal, *S)
- SolverOutputs(*y, *S)

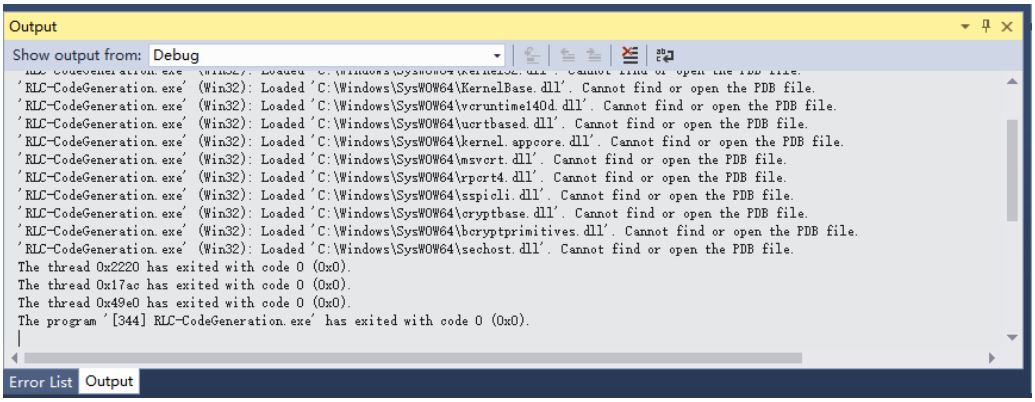
其中 u 是输入信号，p 是子系统参数（例如定义在子系统中的变量），ic 是初始条件，y 是输出，t0 是初始时间，h 是时间步长。S 是一个 C 结构体，包含状态变量的当前值以及其他一些信息。

当 MapleSim 代码在其他第三方仿真工具中执行时（例如 LabVIEW），函数以下面的顺序执行。

- SolverSetup() 当仿真开始时执行
- SolverStep () 求每个时间步上的积分求解
- SolverUpdate() 和 SolverOutputs() 将输出写到 y

15.5.5 编译代码

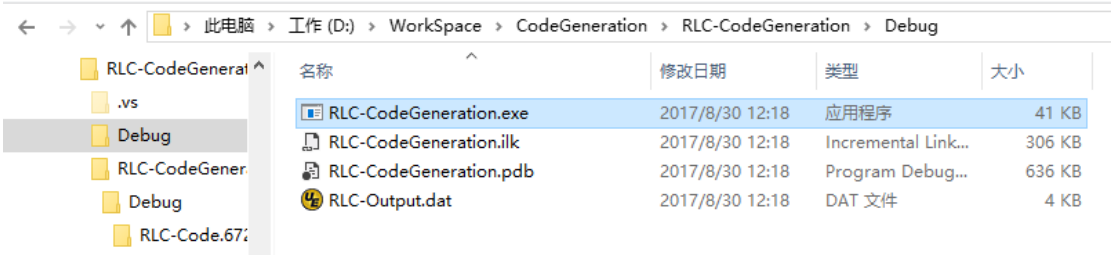
点击 Debug > Starting Debugging。检查 Visual Studio 底部的 output 窗口，将指示 Build 进程是否成功。



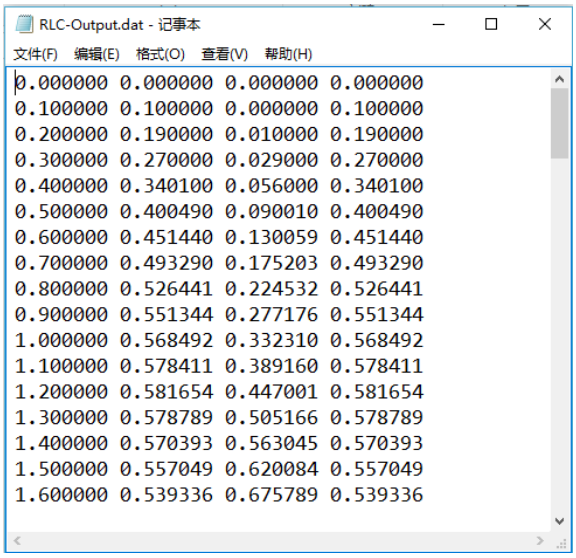
15.6 运行 VC 项目

现在检查项目路径下的 Debug 文件夹，可以找到一个可执行文件。

双击可执行文件，运行它，可以发现新增加了一个 output.dat 文件。

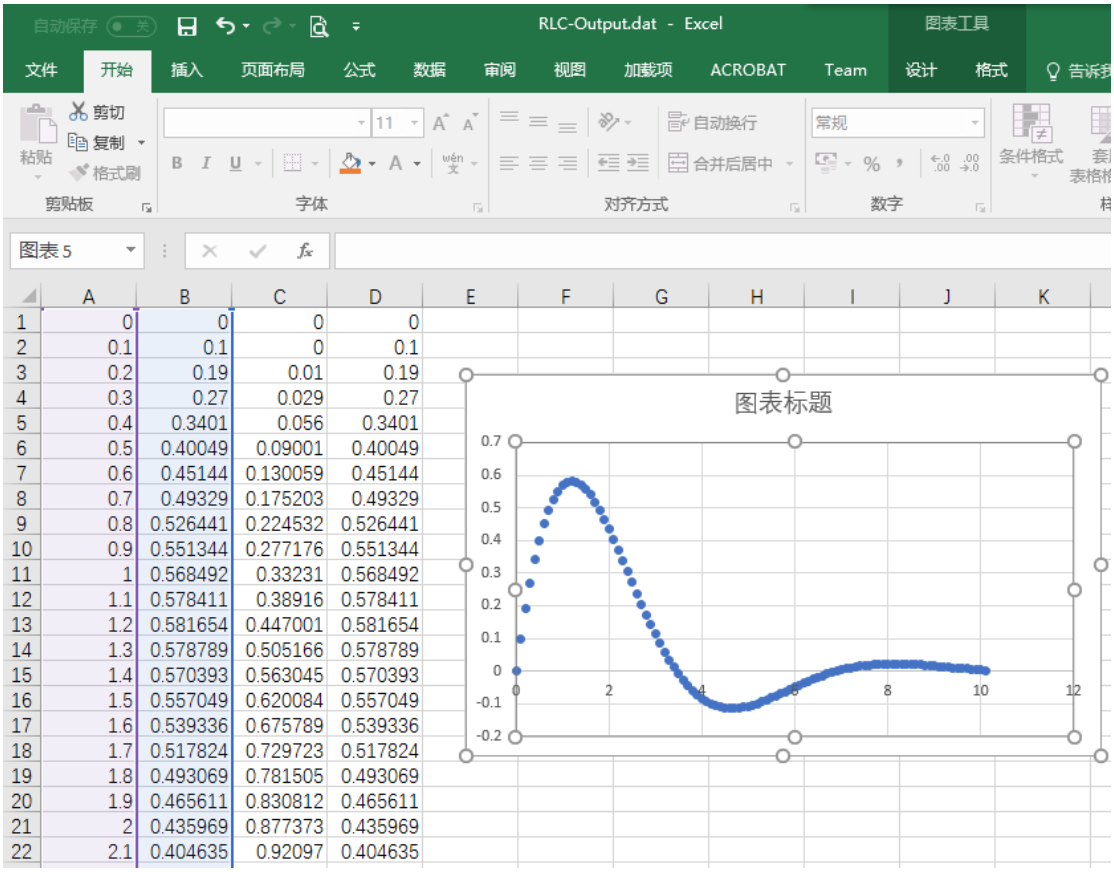


Output.dat 文件包含仿真结果。第一列是仿真时间，后面三列对应是 MapleSim 子系统的输出结果。



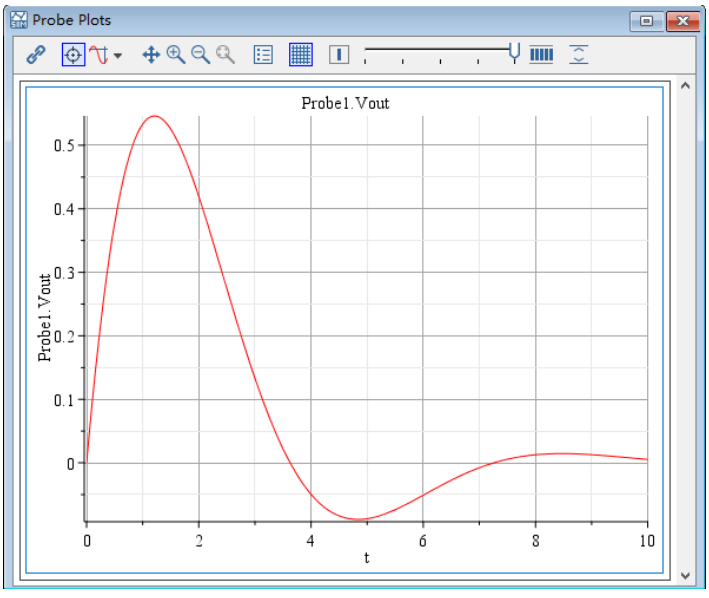
15.7 检查结果

使用 Excel 文件打开 output.dat 文件，并画出第 1 列和第 2 列的图形。如下入所示。



	A	B	C	D
1	0	0	0	0
2	0.1	0.1	0	0.1
3	0.2	0.19	0.01	0.19
4	0.3	0.27	0.029	0.27
5	0.4	0.3401	0.056	0.3401
6	0.5	0.40049	0.09001	0.40049
7	0.6	0.45144	0.130059	0.45144
8	0.7	0.49329	0.175203	0.49329
9	0.8	0.526441	0.224532	0.526441
10	0.9	0.551344	0.277176	0.551344
11	1	0.568492	0.33231	0.568492
12	1.1	0.578411	0.38916	0.578411
13	1.2	0.581654	0.447001	0.581654
14	1.3	0.578789	0.505166	0.578789
15	1.4	0.570393	0.563045	0.570393
16	1.5	0.557049	0.620084	0.557049
17	1.6	0.539336	0.675789	0.539336
18	1.7	0.517824	0.729723	0.517824
19	1.8	0.493069	0.781505	0.493069
20	1.9	0.465611	0.830812	0.465611
21	2	0.435969	0.877373	0.435969
22	2.1	0.404635	0.92097	0.404635

打开 MapleSim 中的 RLCcircuit 电路模型，用一个 constant（k=1）元件替换原有的 Sine 元件，运行仿真。



粗略对比后可以发现 VC 的执行结果和 MapleSim 的仿真结果一致，初步判定模型 C 代码和 VC++项目是正确的。