

自动控制系统仿真

UM 软件入门系列教程
(03)

四川同算科技有限公司 译

2021 年 3 月

前言

本教程介绍使用**UM**软件与**Matlab**软件进行联合仿真的两种方法及基本流程。一种是从**Matlab/Simulink**导出控制系统模型到**UM**软件，另一种是从**UM**软件导出机械系统模型到**Matlab/Simulink**软件。

请读者在学习本课程之前务必先学习《**UM**软件入门系列教程01：多系统动力学仿真》，并熟悉**UM**软件的基本操作：新建模型，创建几何图形、刚体、铰和力元。

本教程将通过两个例子讲解机械系统与控制系统联合仿真的方法。第一个是倒立摆模型，通过控制系统实现稳定倒立；第二个是考虑电机驱动的传动轴模型，模拟电机从启动到匀速工作的过程，并输出电流、电压、电磁力矩和角速度等时程曲线。

首先分别在**UM**和**Matlab/Simulink**里创建机械系统和控制系统，最后将二者耦合起来。这种技术可以帮助我们模拟非常复杂的系统，如汽车**ABS**系统、磁悬浮列车。基于**UM**和**Matlab**实现耦合的方式有两种：第一种是从**Matlab/Simulink**输出动态链接库**DLL**文件到**UM**模型，称之为**Matlab Import**；第二种是从**UM**输出动力学模型到**Matlab/Simulink**作为**S**函数，称之为**Cosimulation**。

请读者务必逐页阅读、一步一步操作，有些基本的软件操作在后面不会详细介绍甚至忽略。

兼容性

UM Control/Matlab Import

Matlab Import是**UM Control**模块的一个子模块，支持从**Matlab/Simulink**导出**DLL**文件至**UM**。请先运行**UM Input**或**UM Simulation**程序，选择菜单**Help | About**，在弹出窗口查看**UM Control/Matlab Import**一栏是否为“+”标记，若显示为“-”，则请重新申请试用或购买正版许可。

目前，**UM Control/Matlab Import**支持64位的**Matlab (R2013-R2020)**版本（**UM**软件自8.3.3.4版本起停止发布32位安装程序）。

为了完成编译**Matlab/Simulink**模型，需要具备**Matlab Coder**和**Simulink Coder**两个工具包。读者可以在**Matlab**命令窗口使用**ver**命令查看。

另外，还必须安装有至少一个**Matlab**支持的微软编译器，可在**Matlab**官方网站查询：

<https://www.mathworks.com/support/requirements/supported-compilers.html>

<https://www.mathworks.com/support/requirements/previous-releases.html>

请注意：联合仿真只支持**Matlab/Simulink**里的连续单元，不支持离散单元。

UM Control/CoSimulation

CoSimulation也是**UM Control**模块的一个子模块，支持从**UM**导出动力学模型至**Matlab/Simulink**。**CoSimulation**执行计算时还会用到**UM COM Server**模块。

目前，**UM Control/CoSimulation**支持**64位**的**Matlab (R2013-R2020)**版本（*UM*软件自**8.3.3.4**版本起停止发布**32位**安装程序）。

版权和商标

本教程仅供读者参考，不同的版本其界面可能有个别不同之处，我们会不定期进行修订。对于本文档中可能出现的任何错误，我们不承担任何责任或义务。

版权所有© 2021 Computational Mechanics Ltd.

俄罗斯计算力学有限公司保留所有权利。

联系方式

最新版的UM软件和相应的用户手册下载地址：

<http://www.universalmechanism.com/en/pages/index.php?id=3>.

若无法访问，请点击：<http://www.umlab.ru/en/pages/index.php?id=3>.

在使用过程中，读者如有任何报错、疑问和建议，请发送邮件至：

um@universalmechanism.com

UM总部

Computational Mechanics Ltd.

Vostochnaya str. 2-14, Glinischevo, Bryansk region, 241525, Russia

Phone, fax: +7 4832 568637

www.universalmechanism.com www.umlab.ru

UM中国

四川同算科技有限公司

四川省眉山市彭山区蔡山西路2号伟业广场1911室

办公电话：028-38520556

公司网站：www.tongsuan.cn

电子邮件：um@tongsuan.cn



微信公众号



QQ 交流群

目 录

1.	UM CONTROL 模块.....	1
1.1	MATLAB IMPORT 工具.....	1
1.2	CoSIMULATION 工具.....	1
2.	模型简介	2
2.1	倒立摆	2
2.1.1	模型简介.....	2
2.1.2	准备 UM 模型.....	3
2.2	直流电机	4
2.2.1	模型简介.....	4
2.2.2	机械系统.....	6
2.2.3	电路系统.....	7
2.3	限制条件	7
3.	使用 MATLAB IMPORT 工具.....	8
3.1	工作流程	8
3.2	倒立摆	9
3.2.1	从 Matlab/Simulink 输出模型.....	9
3.2.2	将 DLL 文件导入 UM.....	17
3.2.3	运动仿真.....	22
3.3	直流电机	24
3.3.1	Matlab/Simulink 模型.....	24
3.3.2	从 Matlab/Simulink 输出模型.....	26
3.3.3	将 DLL 文件导入 UM.....	27
3.3.4	运动仿真.....	30
4.	使用 CoSIMULATION 工具	32
4.1	工作流程	32
4.2	倒立摆	33
4.2.1	准备 Matlab/Simulink 模型	34
4.2.2	输出 UM 模型.....	35
4.2.3	连接 UM 模型和 Matlab/Simulink 模型	40
4.2.4	运动仿真.....	41
4.3	直流电机	43
4.3.1	准备 Matlab/Simulink 模型	43
4.3.2	输出 UM 模型.....	44
4.3.3	连接 UM 模型和 Matlab/Simulink 模型	47
4.3.4	运动仿真.....	48

1. UM Control 模块

UM 提供了多个工具实现机械系统与控制系统的联合仿真，其中与 Matlab/Simulink 相关的有两个。

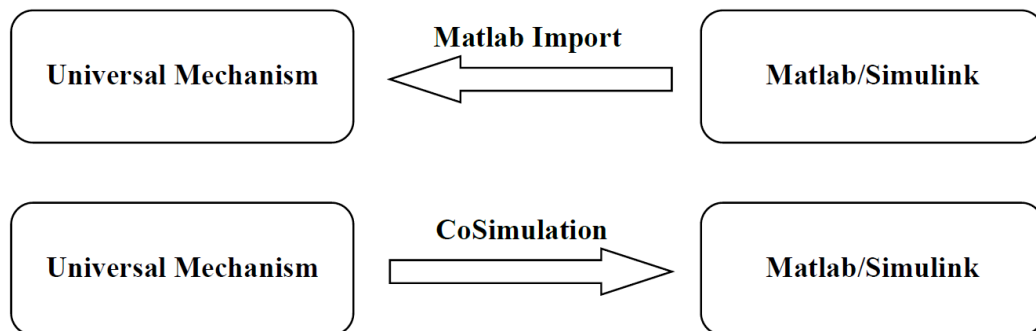
1.1 Matlab Import 工具

先在 Matlab/Simulink 中建立控制系统模型，并编译输出动态链接库（DLL 文件），然后通过 UM 的外部库向导 Wizard of external libraries 导入 UM，在 UM 里进行计算。

1.2 CoSimulation 工具

先在 UM 中建立机械系统模型，并通过专用工具 Wizard of export 生成 m 文件，导入 Matlab/Simulink 作为一个 S 函数，在 Matlab 里进行计算。

下面的框图显示了模型传递方向和执行仿真的主程序。实际上，对于两种方法，数据交互都是双向的。



2. 模型简介

在这一章，我们先介绍倒立摆和直流电机的机械系统模型及其参数。

2.1 倒立摆

2.1.1 模型简介

倒立摆模型如图 2.1 所示。该模型由一个小车 **cart** 和一个单摆 **pendulum** 组成。模型参数见表 2.1。机械系统的输出信号为角度 ψ ，控制系统的输出信号为力 **Force**。随角度变化而变化的控制力使得单摆能保持倒立平衡状态。

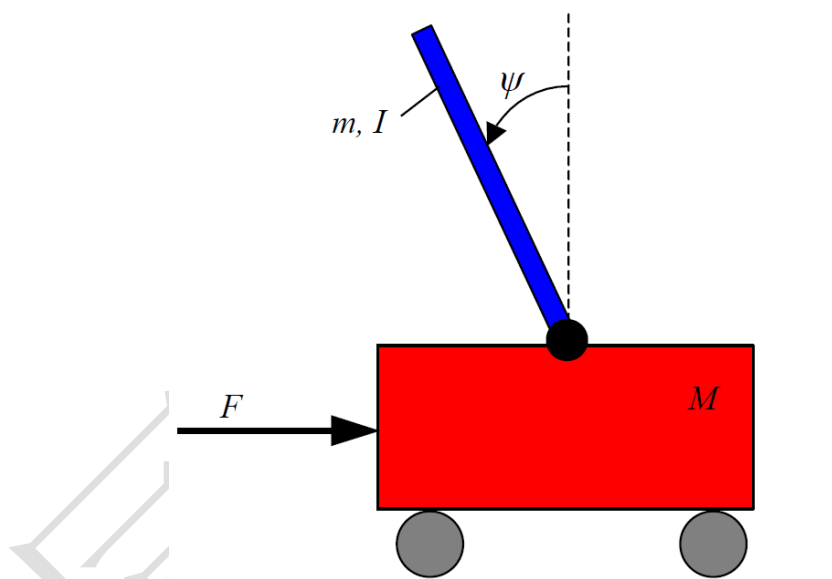


图 2.1 倒立摆

表 2.1 模型参数

参数	模型里的参数符号	备注	数值
F	force	作用在小车上的力	
M	mass_cart	小车的质量	0.5 kg
I	mass_pend	单摆的质量	0.2 kg
	ix	单摆绕转动轴的转动惯量	0.006 kg*m*m
	l	单摆质心到转动轴的距离	0.3 m
	b	小车与地面的摩擦阻尼系数	0.1 Ns/m
ψ		单摆竖向摆动的角度	rad

2.1.2 准备 UM 模型

虽然 **Matlab/Simulink** 动态链接库是在仿真阶段才导入 **UM Simulation** 程序的，但我们必须在建模时就做好准备工作。

如果控制系统通过计算输出力或力矩给机械系统，我们必须先运行 **UM Input** 程序，在机械系统里先定义这么一个力元，并赋予参数符号。

对于本例的倒立摆模型，我们添加了一个 **T-force** 类型的力元。这个力作用于小车上，方向为小车局部坐标系的 **Y** 向，命名为 **force**，如图 2.2 所示。在之后仿真时，使用外部库向导可以将控制系统计算得到的力传递给 **force**。

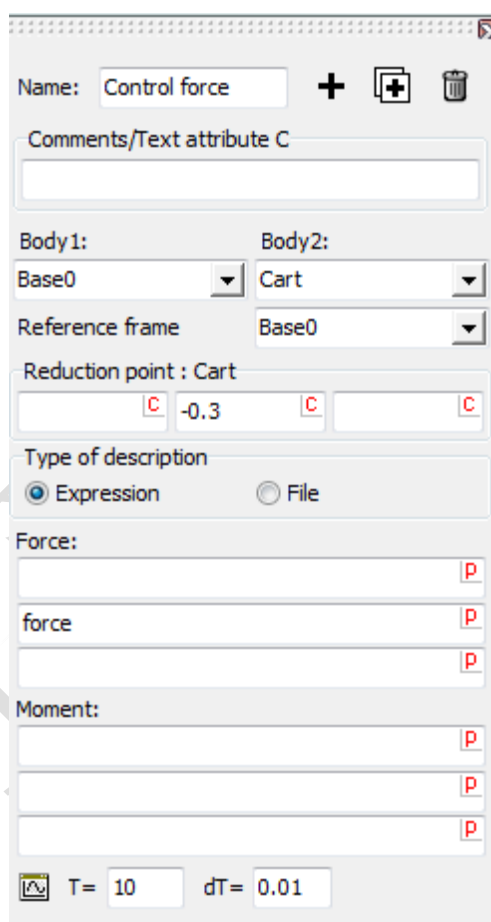


图 2.2 用于连接控制系统的力元

2.2 直流电机

2.2.1 模型简介

本例模型机械系统的主要元件是一个转动惯量为 J_0 的传动轴，如图 2.3 所示。传动轴相对于大地（惯性系）有且仅有一个转动自由度。

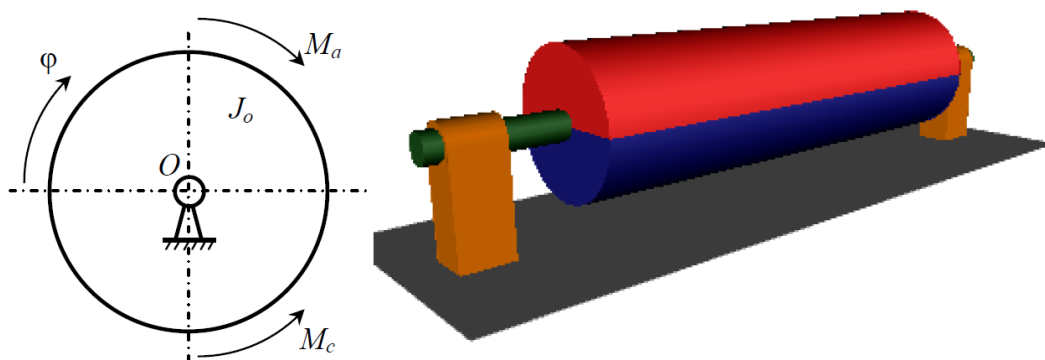


图 2.3 模型简图

模型的电路系统在 **Matlab/Simulink** 里建立，我们假设电机的转子和传动轴是刚性连接的。

电机有一个输入信号和三个输出信号。

输入信号：

- 传动轴（电机转子）的角速度

输出信号：

- 电磁驱动力矩
- 电路中的电流
- 电路中的电压

电磁驱动力矩 M_a 是在 **UM** 模型里定义的 **T-force** 力元，而阻力矩 M_r 则定义为传动轴转动铰的阶跃函数，如图 2.4 所示。

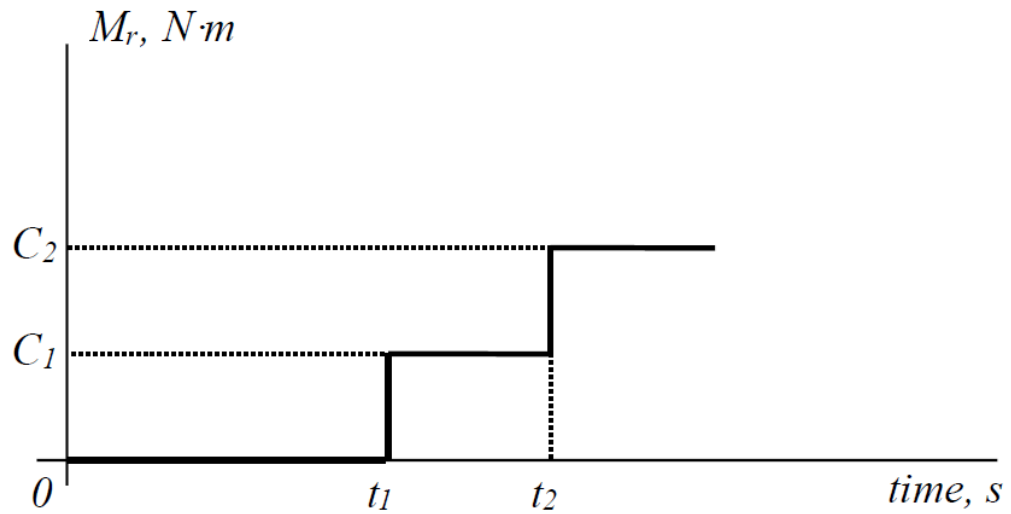


图 2.4 用阶跃函数描述的阻力矩

表 2.2 模型参数

参数	模型里的参数符号	备注	数值
M_a	Driving_torque	电磁驱动力矩	
M_c		阻力矩	F(t)
J_o	Iy	轴的转动惯量	0.1 kg*m*m
t_1	t1	图 2.4 中的时刻 t_1	5 s
t_2	t2	图 2.4 中的时刻 t_2	8 s
C_1	C1	图 2.4 中的阻力矩 C_1	5 N*m
C_2	C2	图 2.4 中的阻力矩 C_2	10 N*m

2.2.2 机械系统

如前所述，为了能连接控制系统，我们需要在机械系统里预先创建一个参数化的力元。

这里，我们打开 **UM Input** 程序，添加一个 **T-force** 力元，命名为 **Driving torque**，如图 2.5 所示。这个 **T-force** 力元有且仅有一个分量 **Ma**，表示绕 **Y** 轴转动的力矩，用于连接控制系统。

转动铰处的阻力矩表示如下：

$$M_r = c_1 * \text{heavi}(t-t_1) + (c_2 - c_1) * \text{heavi}(t-t_2)$$

其中， t_1 ， t_2 ， c_1 和 c_2 是模型参数，如图 2.4 所示。

请注意，**heavi** 函数定义如下：

$$\text{heavi}(t) = \begin{cases} 1, & t > 0 \\ 0, & t \leq 0 \end{cases}$$

为了在仿真时能观察电流和电压的时程曲线，我们可以添加两个参数符号：**I**(电流)和 **U**(电压)，如图 2.5 所示。在仿真时，我们将 **Matlab/Simulink** 模型的另外两个输出信号分别指定给这两参数。

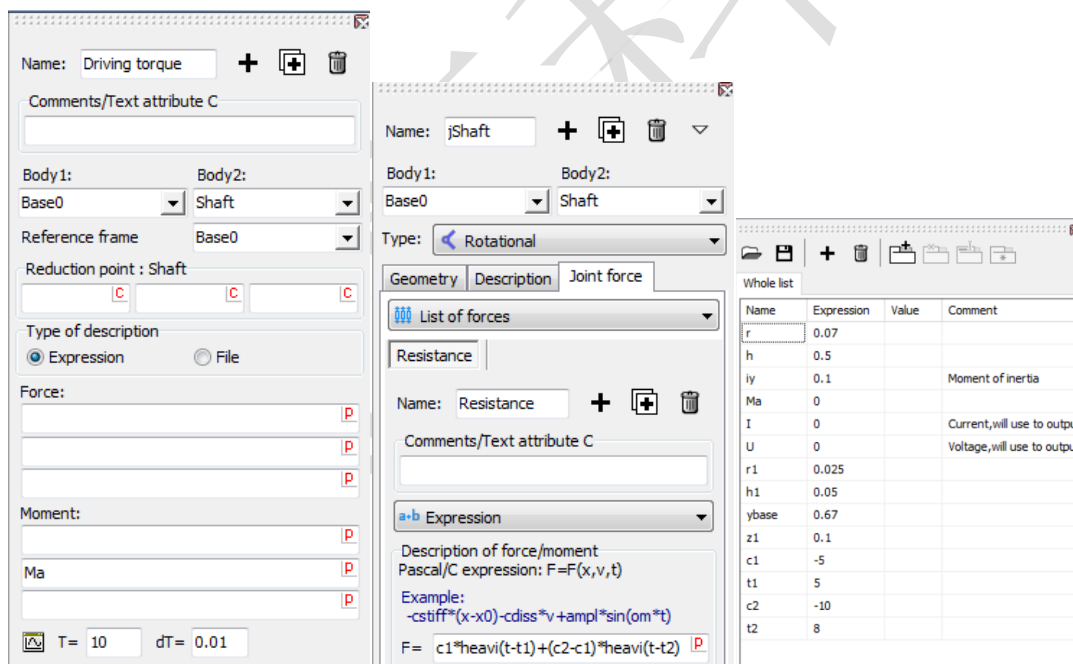


图 2.5 驱动力矩、阻力矩和参数列表

2.2.3 电路系统

在本例中，我们将定义一个恒定磁通量的他励直流电机，其参数见表 2.3。

表 2.3 直流电机参数

参数	数值
额定功率 (kW)	3.6
额定电压 (V)	220
额定转速 (r.p.m)	3000
最大转速 (r.p.m)	4000
效率 (%)	79
15°C 下的电阻 (Ohm)	
电枢回路电阻	0.42
附加电极电阻	0.356
励磁回路电阻	129
电枢回路电感 (mHn)	4.8

2.3 限制条件

Matlab/Simulink 与 **UM** 联合仿真时，存在一些限制条件：可以将 **Matlab/Simulink** 的输出信号传递给力元相关的符号：如力的作用点、刚度和阻尼系数。

根据 **UM** 软件的基本原则，不支持将 **Matlab/Simulink** 的输出信号传递给刚体的质量惯量参数、几何参数和铰坐标，因为这些参数只能在每次仿真开始前设置，而不能在仿真过程中随时间不断变化。

3. 使用 Matlab Import 工具

3.1 工作流程

从 **Matlab/Simulink** 导出控制系统到 **UM** 进行联合仿真的流程如下：

- 在 **Matlab/Simulink** 里搭建控制框图；
- 从 **Matlab/Simulink** 输出动态链接库（**DLL** 文件）；
- 在 **UM Input** 程序里创建机械系统模型；
- 在 **UM Simulation** 程序里加载机械系统模型，利用外部库向导（**Wizard of external libraries**）导入 **Matlab/Simulink DLL** 文件，并设置好连接；
- 进行动力学仿真。

UM 将 **Matlab/Simulink** 模型考虑为一个具有若干个输入和输出信号的黑盒子。输入信号为 **UM Simulation** 程序里变量向导（**Wizard of variables**）创建的变量；输出信号为 **UM Input** 程序里创建的参数符号。

为了实现力或力矩的主动控制效果，必须在 **UM** 里创建相关力元，并以参数化描述，这样才能与 **Matlab/Simulink** 的输出信号建立连接。

3.2 倒立摆

UM 软件自带的包含控制系统的倒立摆模型位于本地目录 {UM Data}\SAMPLES\TUTORIAL\inv_pend。在开始学习本课程之前，请先确认这个模型是否存在。如果没有找到，可以从 UM 软件官方网站下载：
http://www.umlub.ru/download/90/inv_pend.zip

我们将直接使用它，这里就不再详细介绍其建模过程，而着重讲解有关连接 Matlab/Simulink 模型的方法。

包含控制系统的 UM 模型位于本地目录 {UM Data}\SAMPLES\simulink\inv_pend_ctrl。

3.2.1 从 Matlab/Simulink 输出模型

从 Matlab/Simulink 输出控制系统模型意味着将模型编译为动态链接库，用于连接 UM 模型。

备注：在学习之前，最好先熟悉 Matlab/Simulink 的基本操作，以保证能顺利地编译成功 DLL 文件。不推荐直接使用软件自带的动态链接库 pendpid_ctrl.dll 文件，因为它可能是 32 位版本 Matlab/Simulink 编译的，不能用于 64 位 UM 软件进行仿真。

提示：Matlab/Simulink 里的 **Derivative** 模块不支持导出使用，可用 **Transfer Fnc** 模块代替。

Matlab/Simulink 里的基本流程如下：

1. 准备一个控制系统模型；
2. 复制编译 DLL 所需的文件到模型目录；
3. 设置编译选项并执行编译。

准备控制系统模型

首先，必须去掉 **Matlab/Simulink** 模型中不能被编译的模块/组件：所有的输入/输出组件，以及所有没有源代码的组件。

请注意模型必须有“**IN**”和“**OUT**”组件，用于与 **UM** 模型进行连接。在本例中，分别有一个“**IN**”（单摆竖向摆角）接口和“**OUT**”（作用在小车上的控制力）接口。

Matlab/Simulink 控制系统如图 3.1 所示。

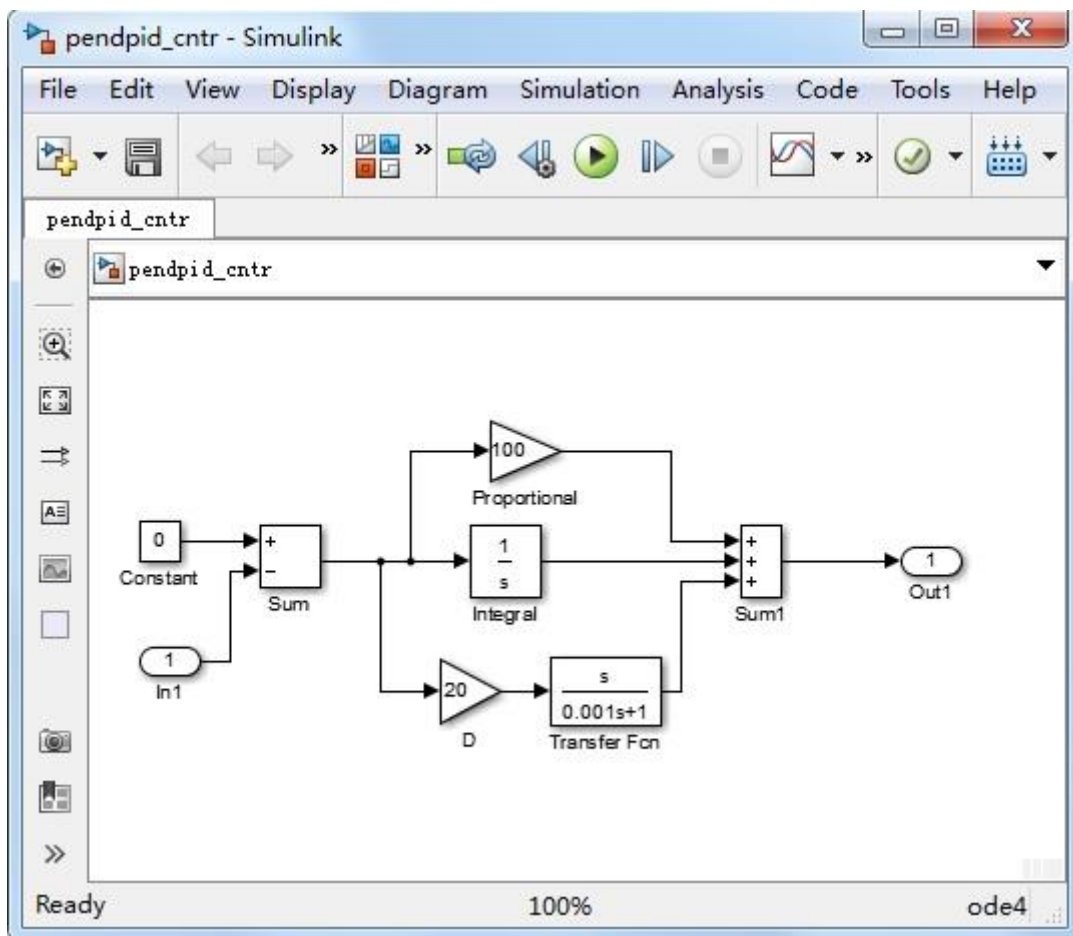


图 3.1 倒立摆模型的控制框图

复制编译所需文件和设置工作目录

1. 为了将控制模型正确编译成 **UM** 所需的 **DLL** 文件，我们需要先从目录 **{UM Data}\Simulink** 复制相应版本的接口文件到存放 **Matlab/Simulink** 模型的目录下。

Matlab 8.X / R2013-R2015 32 位

复制 {UM Data}\Simulink\R2013_2015\x32 目录下的 rsim.tlc, rsim_vc.tmf 和 um.tls 文件到模型文件夹。

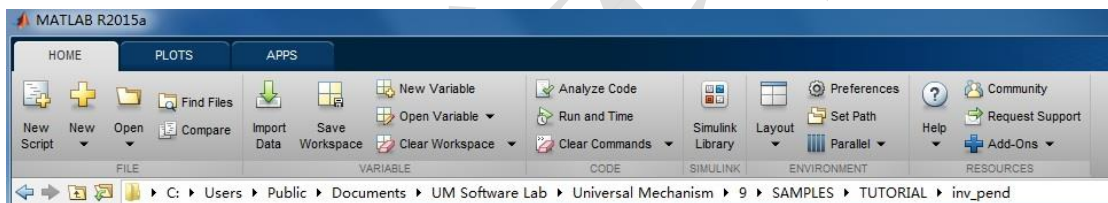
Matlab 8.X / R2013-R2015 64 位

复制 {UM Data}\Simulink\R2013_2015\x64 目录下的 rsim.tlc, rsim_vc.tmf 和 um.tls 文件到模型文件夹。

Matlab 9.X / R2016-R2020 64 位

复制 {UM Data}\Simulink\R2016_R2020\x64 目录下的 rsim.tlc, rsim_vc.tmf 和 um.tls 文件到模型文件夹。

2. 在 **Matlab** 主页窗口置模型目录为当前工作目录。



编译 DLL

1. 在 **Matlab** 命令窗口输入命令：**mex -setup**（注意 *mex* 后有一个空格），然后从列表中选择一个编译器，或直接输入 **mex -setup C++**。

```
Command Window
>> mex -setup C++
MEX configured to use 'Microsoft Visual C++ 2010' for C++ language compilation.
```

备注：我们推荐使用 Microsoft Visual C/C++编译器来编译生成动态链接库。

使用 Matlab 8.X / R2013-R2015 编译 DLL

1. 在 Simulink 模型窗口，选择菜单 **Code | C/C++ Code | Code Generation Options**，弹出界面如图 3.2。
2. 在 **Code Generation** 页面设置 **System target file** 为 **rsim.tlc**。
3. 设置 **Language** 为 **C**。
4. 设置 **Make command** 为 **make_rtw**。
5. 设置 **Template makefile** 为 **rsim_vc.tmf**。
6. 在 **Solver** 页面设置 **Type** 为 **Fixed Step**。
7. 在 **Solver options | Solver** 选择合适的求解器，如 **ode4**（四阶龙格库塔法）。
8. 回到 **Code Generation** 页面，点击 **Build**，编译生成 **DLL** 文件。

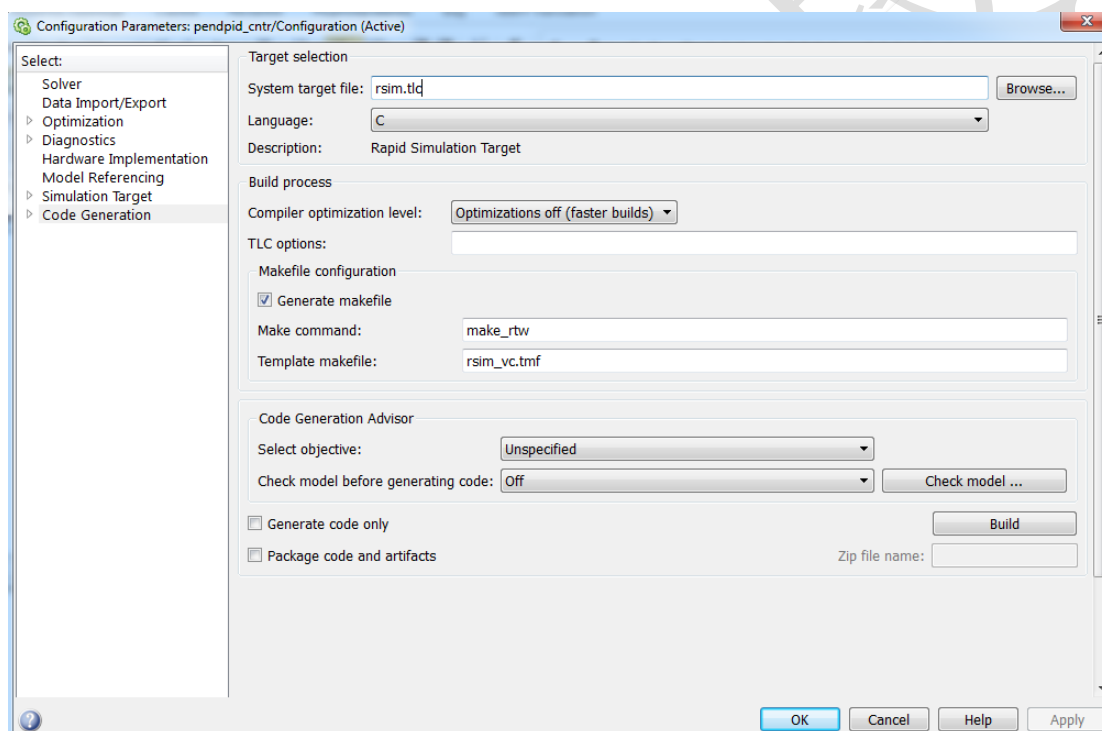


图 3.2 设置编译选项（Matlab 8.X R2013-R2015）

编译成功后，会出现提示“**Creating LIBRARY..\pendpid_cntr.lib and object ..\pendpid_cntr.exp**”。最后得到 **pendpid_cntr.dll** 文件。

备注：为确保正常计算，最好将 Matlab 和 UM 软件安装在同一台计算机。

使用 Matlab 9.X / R2016-R2020 编译 DLL

1. 在 Simulink 界面定位到 `pendpid_ctrl` 模型窗口。
2. 选择 **MODELLING | Model Settings | Model Settings**，或在空白处点击右键，选择 **Model Configuration Parameters** 菜单，弹出窗口如图 3.4。

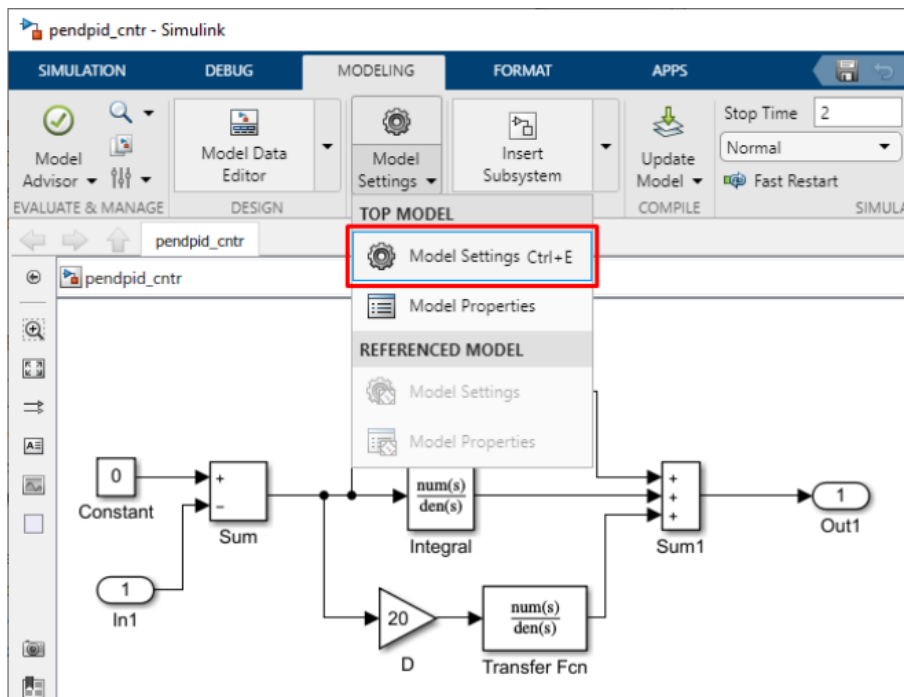


图 3.3 设置编译选项 (Matlab 9.X R2016-R2020)

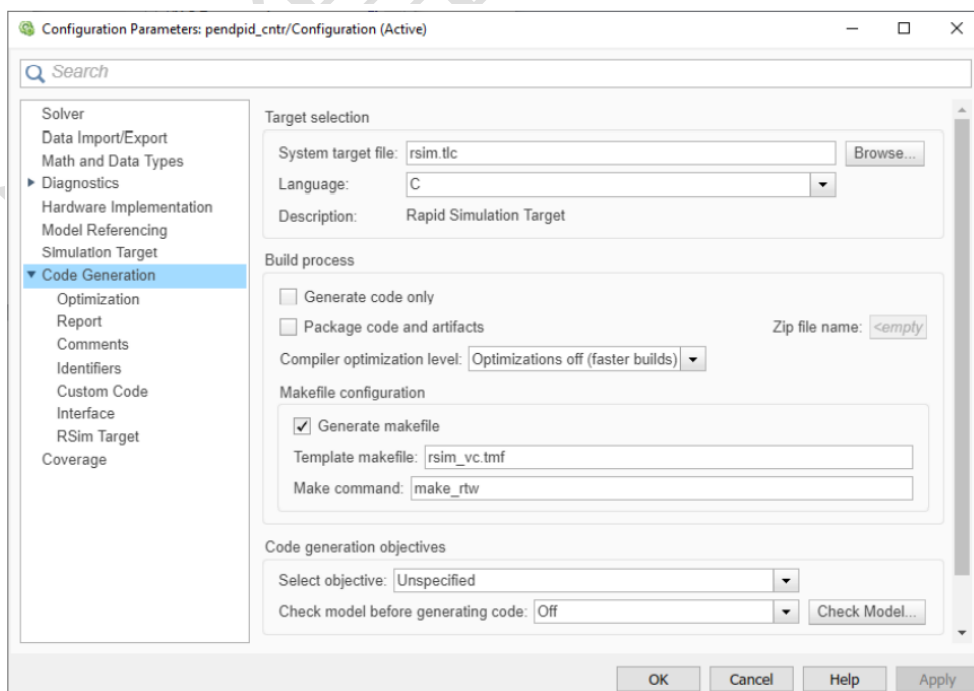


图 3.4

3. 定位到 **Code Generation** 页面。
4. 设置 **System target file** 为 **rsim.tlc**。
5. 设置 **Language** 为 **C**。
6. 设置 **Template makefile** 为 **rsim_vc.tmf**。
7. 设置 **Make command** 为 **make_rtw**。
8. 切换到 **Solver** 页面。
9. 设置 **Solver options | Type** 为 **Fixed Step**。
10. 在 **Solver options | Solver** 选择合适的求解器，如 **ode4**（四阶龙格库塔法）。
11. 然后到 **Code Generation | Optimization** 页面，设置 **Default parameter behavior** 为 **Tunable**，如图 3.5。
12. 关闭 **Configuration Parameters** 窗口。

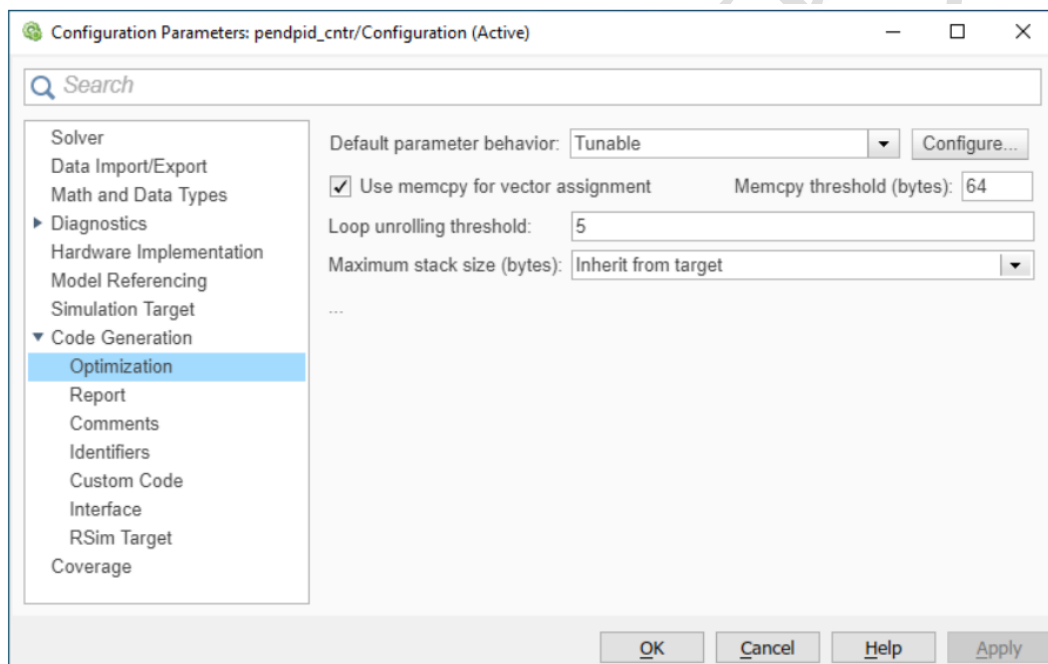


图 3.5 Matlab 9.X 的 Tunable 参数

13. 最后从 **APPS** 里选择 **Simulink Coder | Build (Generate code and build model)**，完成编译，如图 3.6 和图 3.7。

编译成功后，会出现提示“**Build process completed successfully**”。最后得到 **pendpid_cntr.dll** 文件，以及源代码（位于 **pendpid_cntr_rsim_rtw** 目录）。

备注：为确保正常计算，最好将 Matlab 和 UM 软件安装在同一台计算机。

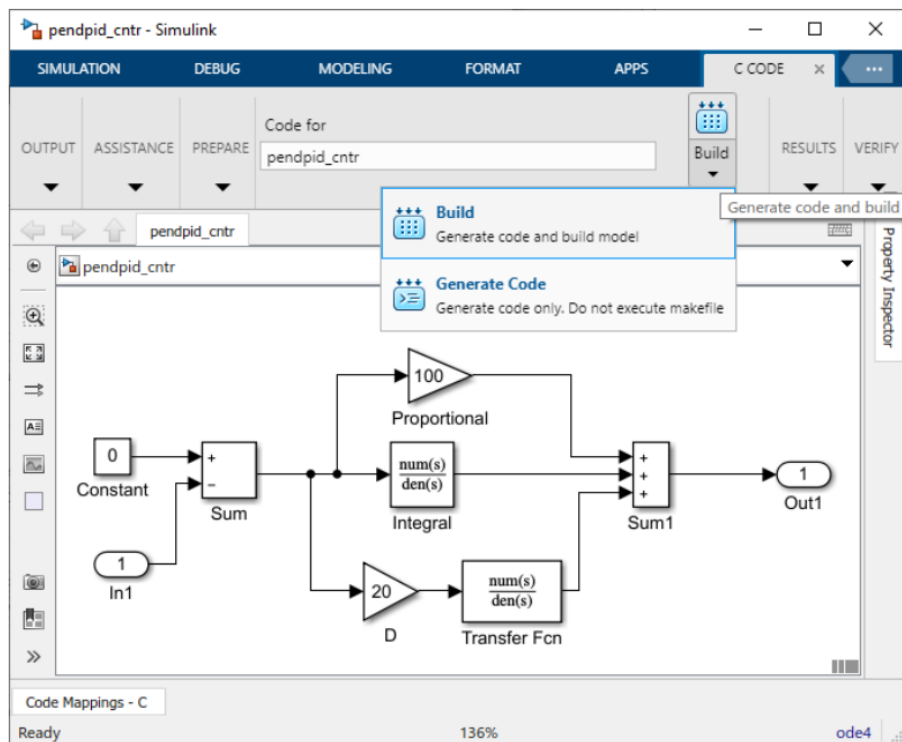


图 3.6 使用 Matlab9.X 进行编译

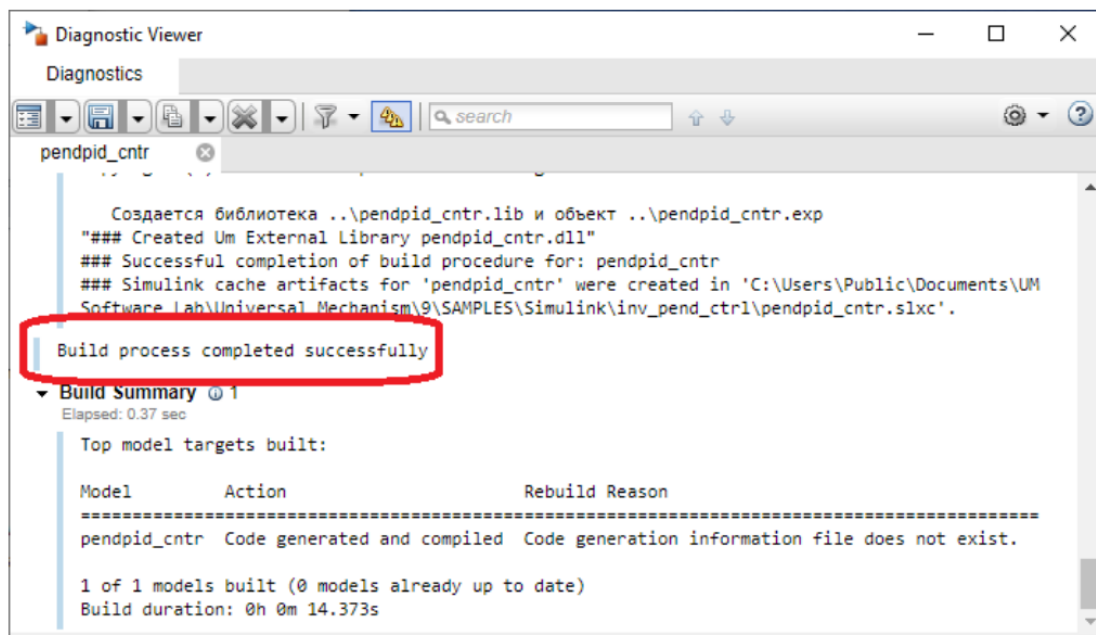


图 3.7 进程查看

3.2.2 将 DLL 文件导入 UM

加载 Matlab/Simulink 动态链接库

1. 运行 **UM Simulation** 程序。
2. 加载模型 {UM Data}\SAMPLES\TUTORIAL\inv_pend。
3. 选择菜单 **Tools | External library interface**，弹出外部库向导窗口。

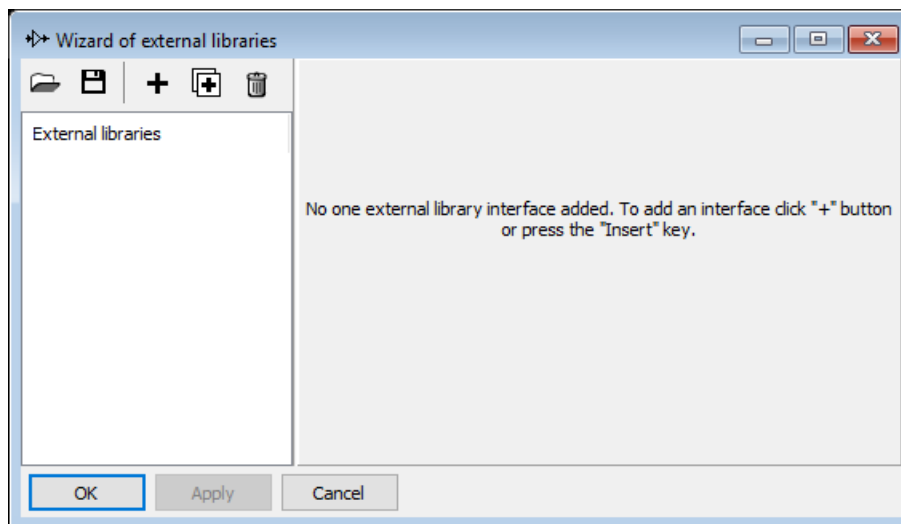


图 3.8 外部库向导

4. 点击按钮 **+**，添加一个外部库。
5. 在右上角 **Path to external library** 处点击按钮 ，选择在 **Matlab** 里编译的动态链接库文件 **pendpid_cntr.dll**，如图 3.9 所示。
6. 在图 3.9 页面左侧，勾选 **Interface0**。

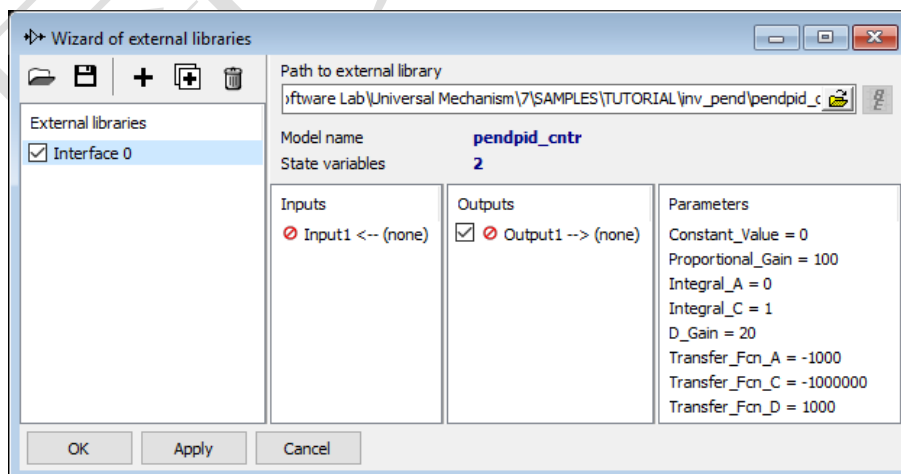


图 3.9 添加一个外部库

外部库向导加载动态链接库时，自动识别出控制系统的输入和输出接口，并列表显示。本例模型只有一个输入和一个输出。

重命名

1. 在外部库向导窗口左侧选中 **Interface0**，点右键，选择菜单 **Rename**。
2. 输入 **Control force**，并回车。

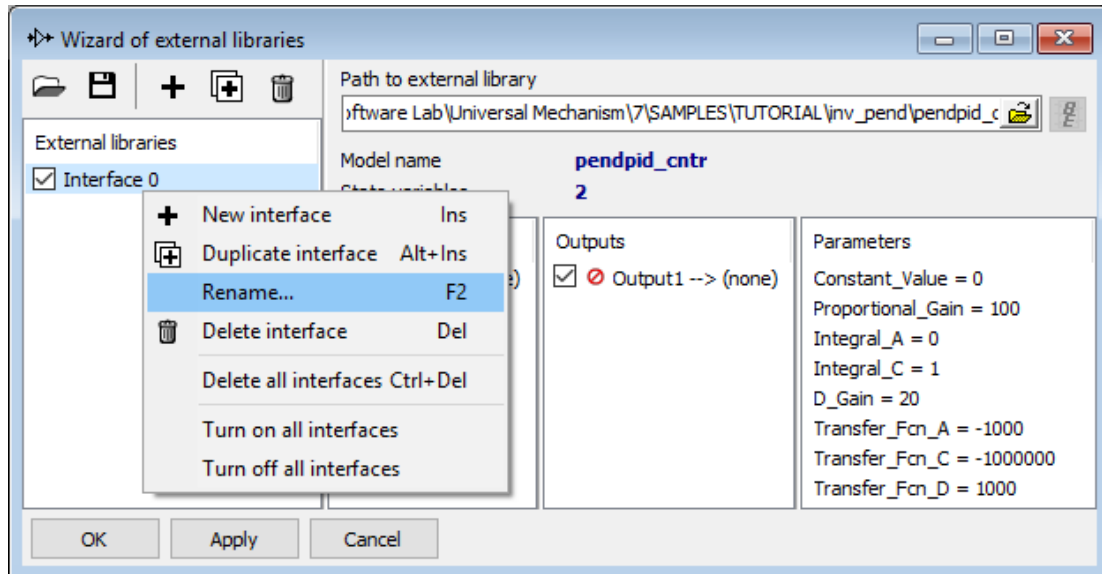



图 3.10 重命名

定义控制系统的输入信号

1. 选择菜单 **Tools | Wizard of variables**，打开变量向导。
2. 选择角度变量 **Angular variables**。
3. 勾选 **Use orientation at zero coordinates**。
4. 在左侧列表选中 **Pendulum**，选择 **Type of variable** 为 **Rot.vector**，设置分量 **Component** 为 **X**，如图 3.11 所示。
5. 点击按钮  创建单摆绕 **X** 轴转动的角度变量。

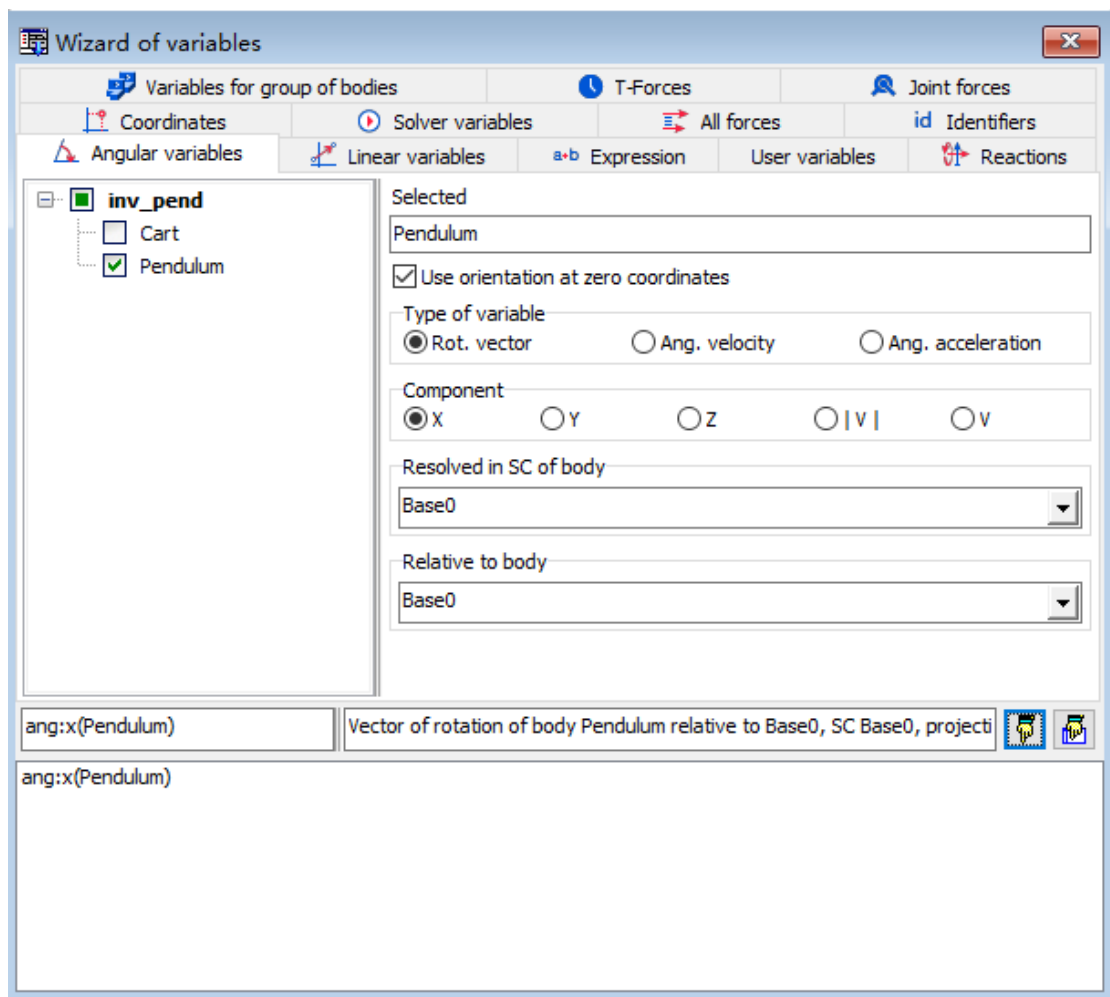


图 3.11 定义角度变量

- 将上一步创建的角度变量 **ang:x(Pendulum)** 从变量向导窗口拖到外部库向导窗口 **Input1** 处，作为控制系统的输入信号，最后如图 3.12 所示。

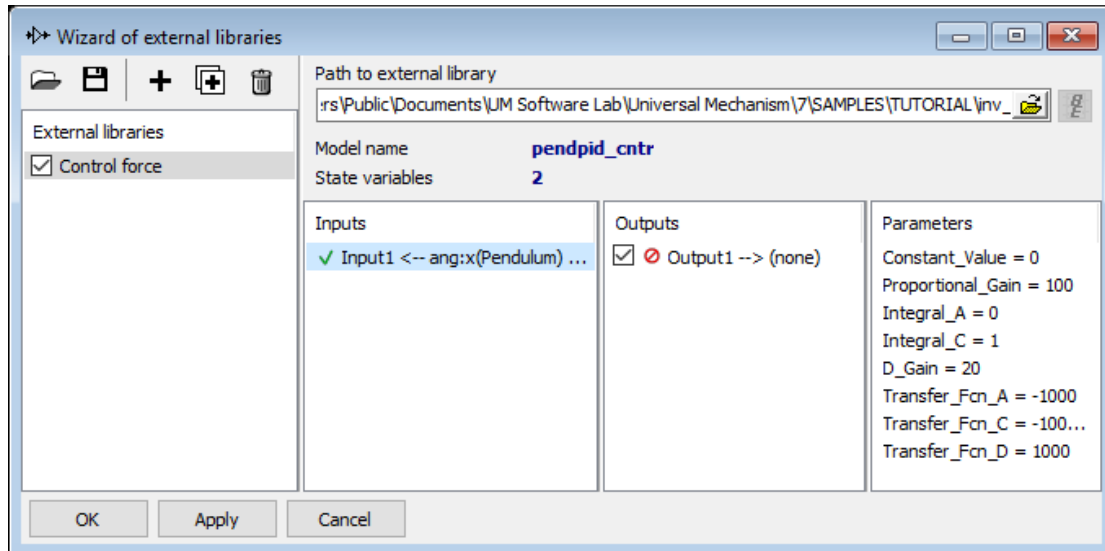


图 3.12 定义控制系统的输入变量

定义控制系统的输出信号

1. 在外部库向导窗口双击 **Output1**，弹出模型参数窗口。
2. 从下拉菜单选择参数 **force**，如图 3.13 所示。
3. 点击 **OK**。

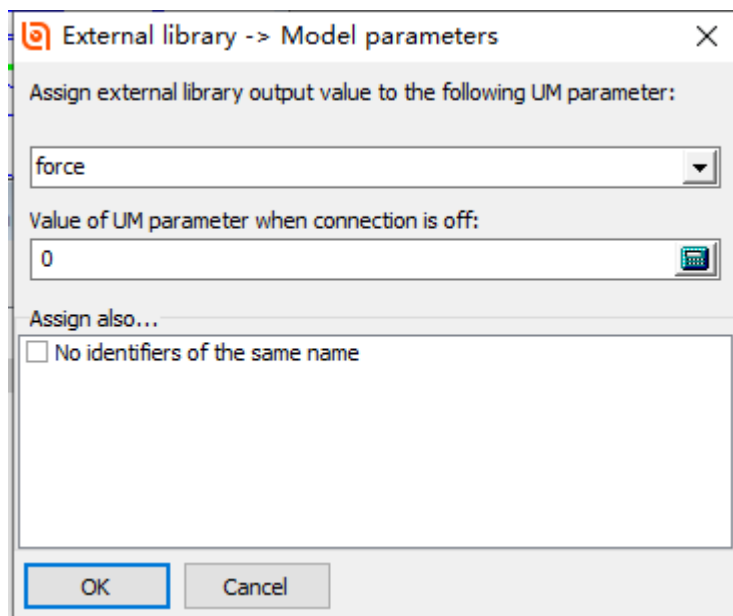


图 3.13 定义输出变量

4. 在图 3.14 所示界面，点击 **Apply**（保存设置但不关闭窗口）或 **OK**（保存设置同时关闭窗口）。

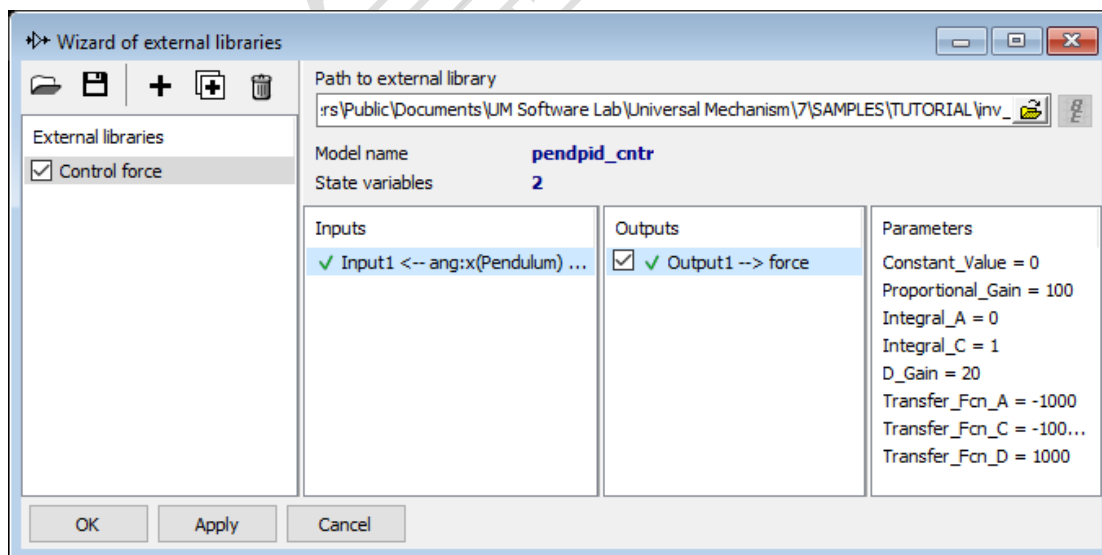


图 3.14 完成输入输出设置的界面

至此，倒立摆模型的机械系统和控制系统已创建连接，可以进行数值仿真了。

3.2.3 运动仿真

1. 选择菜单 **Analysis | Simulation**，弹出仿真控制界面。
2. 点击初始条件 **Initial conditions**。
3. 设置 **Coordinate/1.2** 为 **0.3**，表示将单摆初始转动 **0.3** 弧度，如图 3.15 所示。

备注：坐标 1.1 表示小车的平动自由度，1.3 和 1.4 则为控制系统的状态变量。

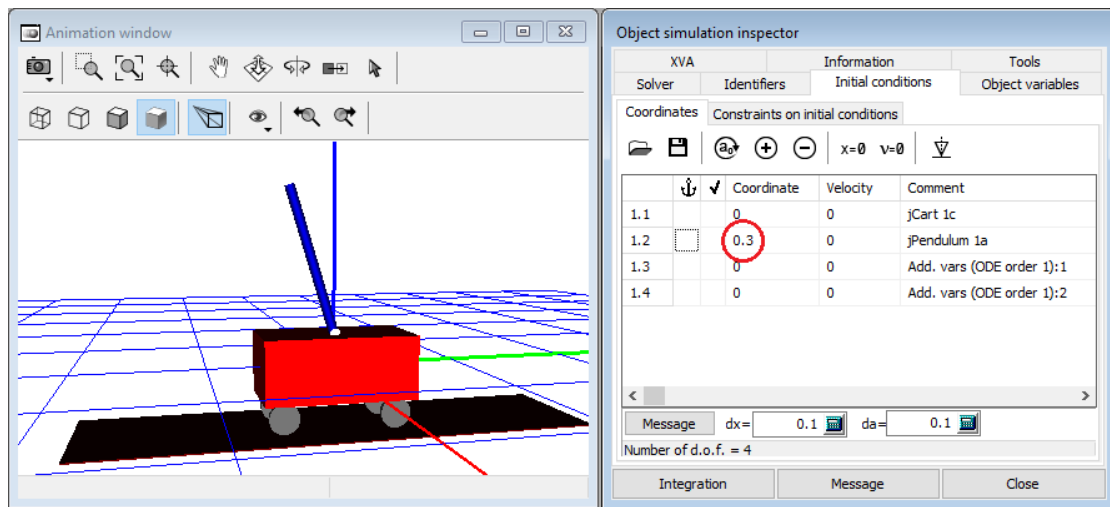


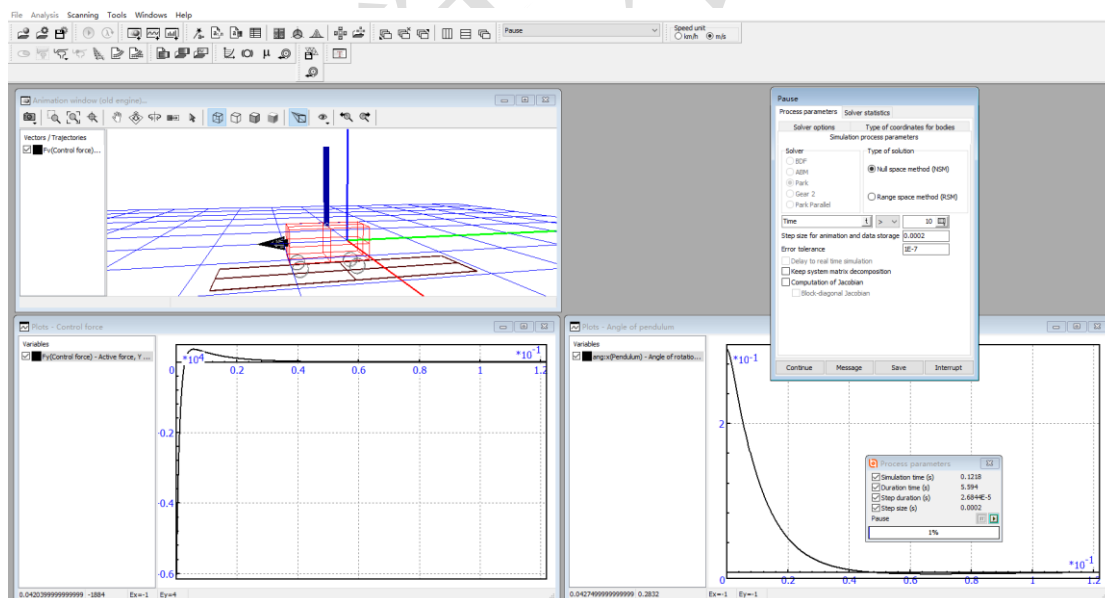


图 3.15 设置初始条件

4. 点击 **Solver**，选择 **Park** 求解器，设置仿真时间为 **0.1s**，数据和动画步长为 **0.0002s**，容差 **Error tolerance** 为 **1E-7**。
 5. 点击 **Integration**，开始仿真，待仿真完成后点击 **Interrupt**。
- 从动画窗口可以看到，单摆在在控制力作用下能保持稳定。

控制力可视化和变量绘图

1. 选择菜单 **Tools | Wizard of variables**，打开变量向导。
2. 点击 **T-Forces**，在左侧列表选中 **Control force**，右侧 **Type of variable** 选择 **Force**，设置分量为 **V**（表示矢量），**Acts on** 作用于 **Body2**（小车）。
3. 点击按钮 ，创建变量，并将其拖入动画窗口。
4. 点击动画窗口工具栏图标 ，模型显示为线框模式。
5. 选择菜单 **Tools | Graphical window**，打开一个绘图窗口。
6. 在变量向导 **T-Force** 页面，设置分量为 **Y**，创建变量，并拖入绘图窗口。
7. 再打开一个绘图窗口。
8. 点击变量向导 **Angular variables** 页面。
9. 左侧选中 **Pendulum**，右侧 **Type of variable** 选择 **Rot.vector**，设置分量 **X**。
10. 创建变量，并拖入到第二个绘图窗口。
11. 关闭变量向导。
12. 开始仿真。



3.3 直流电机

UM 软件自带的包含控制系统的直流电机模型位于本地目录 {UM Data} \SAMPLES\TUTORIAL\dc_motor。在开始学习本课程之前，请先确认这个模型是否存在。如果没有找到，可以从 UM 软件官方网站下载：http://www.umlub.ru/download/90/dc_motor.zip。

我们将直接使用它，这里就不再详细介绍其建模过程，而着重讲解有关连接 Matlab/Simulink 模型的方法。

包含控制系统的 UM 模型位于本地目录 {UM Data} \SAMPLES\simulink\dc_motor_fin。

3.3.1 Matlab/Simulink 模型

本例直流电机 Matlab/Simulink 模型有一个输入信号（角速度）和三个输出信号（电磁力矩、电流和电压），电路控制系统如图 3.16 和图 3.17 所示。

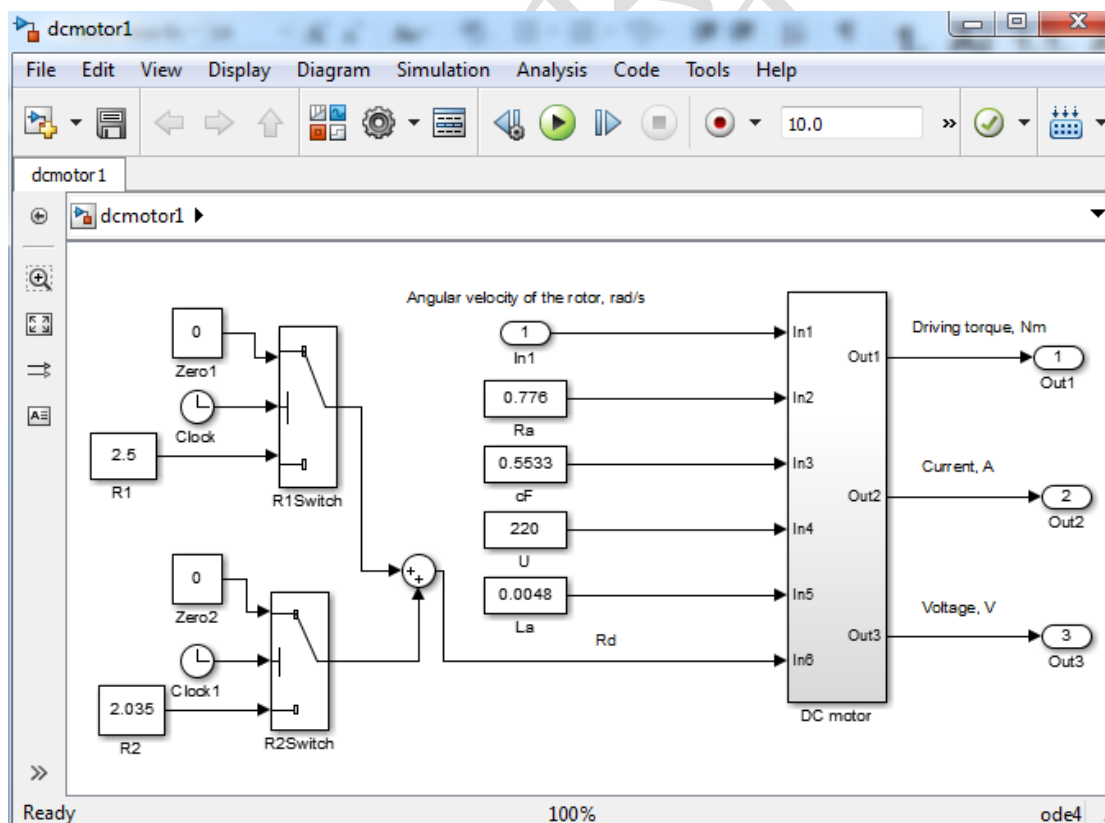


图 3.16 电路系统总图

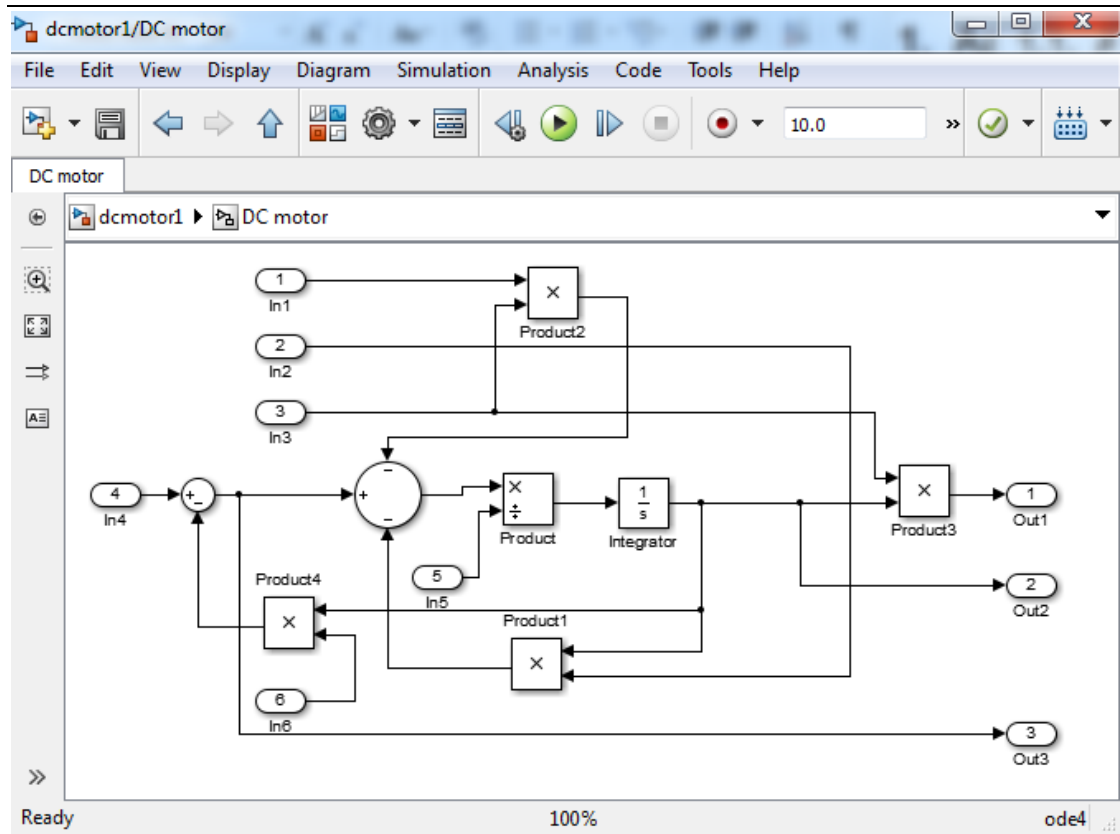


图 3.17 直流电机子系统

3.3.2 从 Matlab/Simulink 输出模型

现在，请参考 3.2.1 章节介绍的方法将“{UM Data} \SAMPLES\TUTORIAL\dc_motor”模型里的 **dcmotor1.mdl** 文件编译为 **dcmotor1.dll**。

不推荐直接使用软件自带的动态链接库 **dcmotor1.dll** 文件，因为它可能是 32 位版本 **Matlab/Simulink** 编译的，不能用于 64 位 **UM** 软件进行仿真。

3.3.3 将 DLL 文件导入 UM

加载 UM 模型

1. 运行 **UM Simulation** 程序。
2. 加载模型 {UM Data}\SAMPLES\TUTORIAL\dc_motor。

加载 Matlab/Simulink 动态链接库

1. 选择菜单 **Tools | External library Interface...**，弹出外部库向导窗口。
2. 点击按钮 **+** 添加一个外部库。
3. 在右上角 **Path to external library** 处点击按钮 ，选择在 **Matlab** 里编译好的动态链接库文件 **dcmotor1.dll**，如图 3.18 所示。
4. 在图 3.18 页面左侧，勾选 **Interface0**。

外部库向导加载动态链接库时，自动识别出控制系统的输入和输出接口，并列表显示。本例模型只有一个输入（轴的角速度）和三个输出（驱动力矩、电流和电压）。

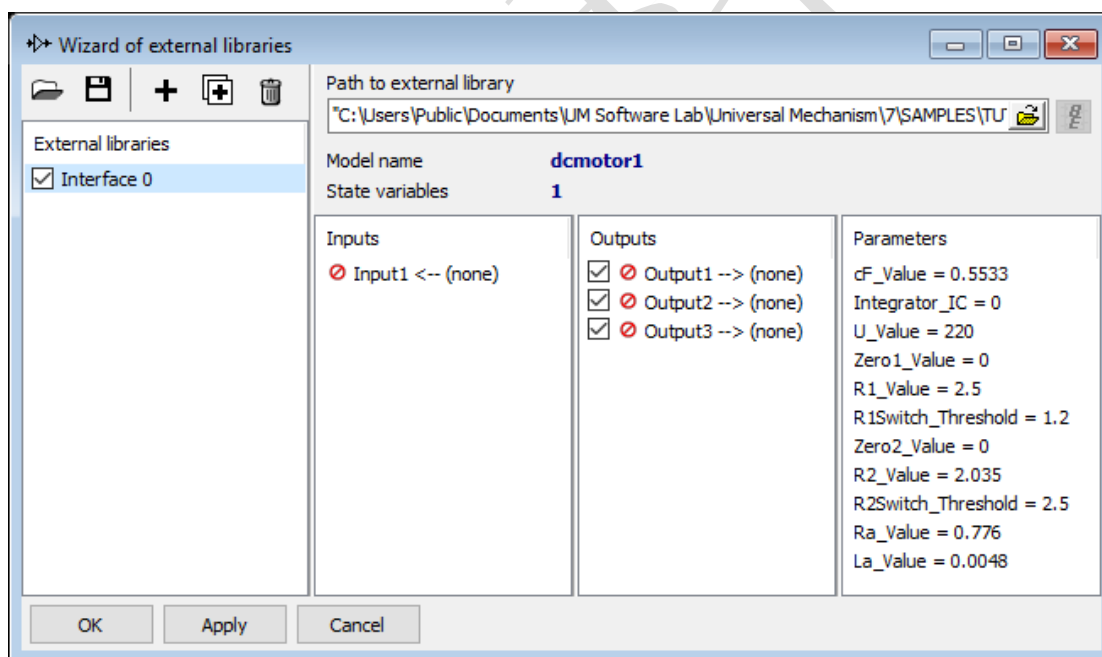



图 3.18 导入 Matlab/Simulink 动态链接库

重命名

1. 在外部库向导窗口左侧选中 **Interface0**，点右键，选择菜单 **Rename**。
2. 输入 **DC Motor**，并回车。

定义控制系统的输入信号

1. 选择菜单 **Tools | Wizard of variables**，打开变量向导。
2. 选择角度变量 **Angular variables**。
3. 在左侧列表选中 **Shaft**，选择 **Type of variable** 为 **Ang.velocity**，设置分量 **Component** 为 **Y**，如图 3.19 所示。
4. 点击按钮  创建传动轴角速度变量 **om:y(Shaft)**。

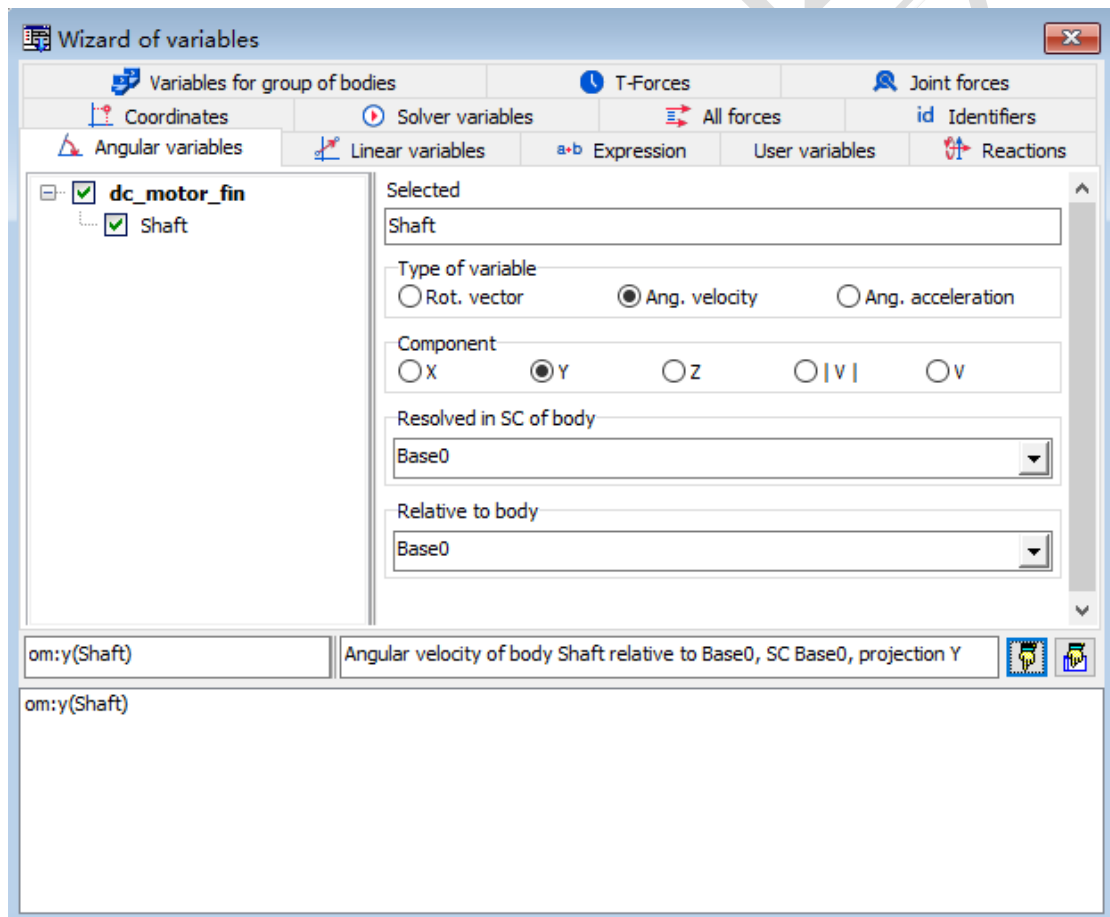


图 3.19 创建传动轴角速度变量

5. 将变量 **om:y(Shaft)**拖到外部库向导窗口 **Input1** 处，作为控制系统的输入信号。
6. 关闭变量向导。

定义控制系统的输出信号

1. 在外部库向导窗口，双击 **Output1**，弹出模型参数窗口。
2. 从下拉菜单选择参数 **Ma**，如图 3.20 所示。
3. 点击 **OK**。

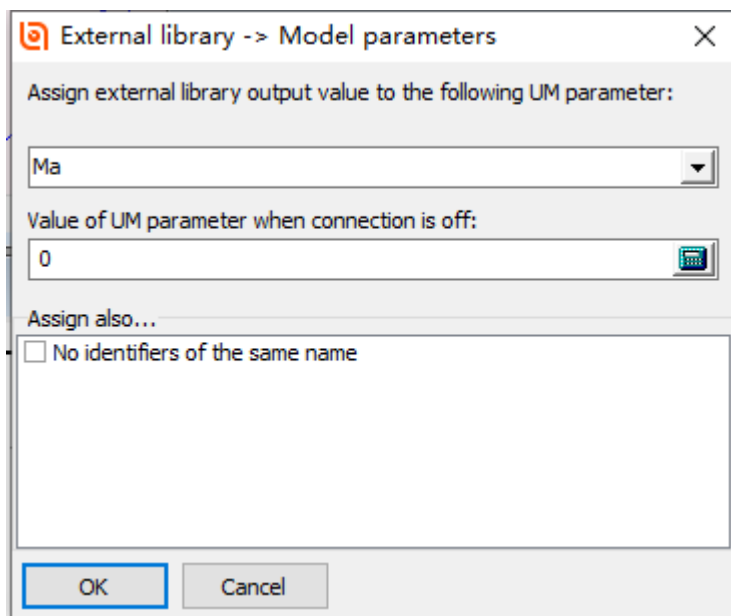


图 3.20

4. 双击 **Output2**，选择参数 **I**，点击 **OK**。
5. 双击 **Output3**，选择参数 **U**，点击 **OK**；最终参数设置如图 3.21 所示。
6. 点击外部库向导窗口 **OK**，保存设置并关闭窗口。

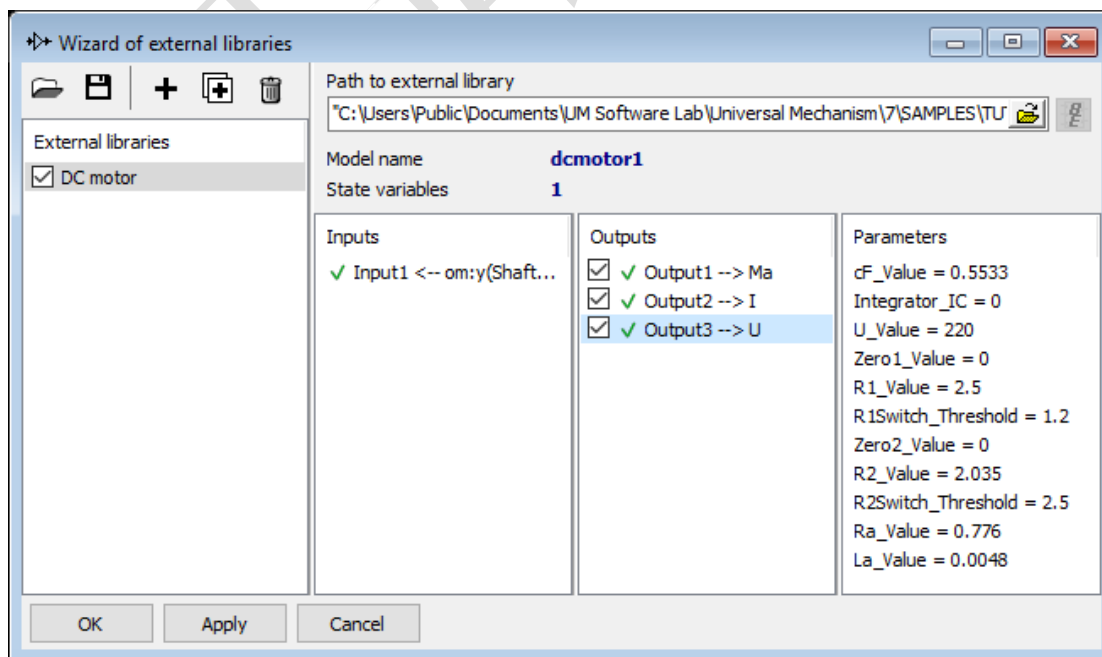


图 3.21

3.3.4 运动仿真

仿真前的准备

1. 选择菜单 **Tools | Animation window**，打开一个动画窗口。
2. 选择菜单 **Tools | Graphic window**，打开一个绘图窗口，再重复两次，最终仿真程序桌面布置如图 3.22 所示。

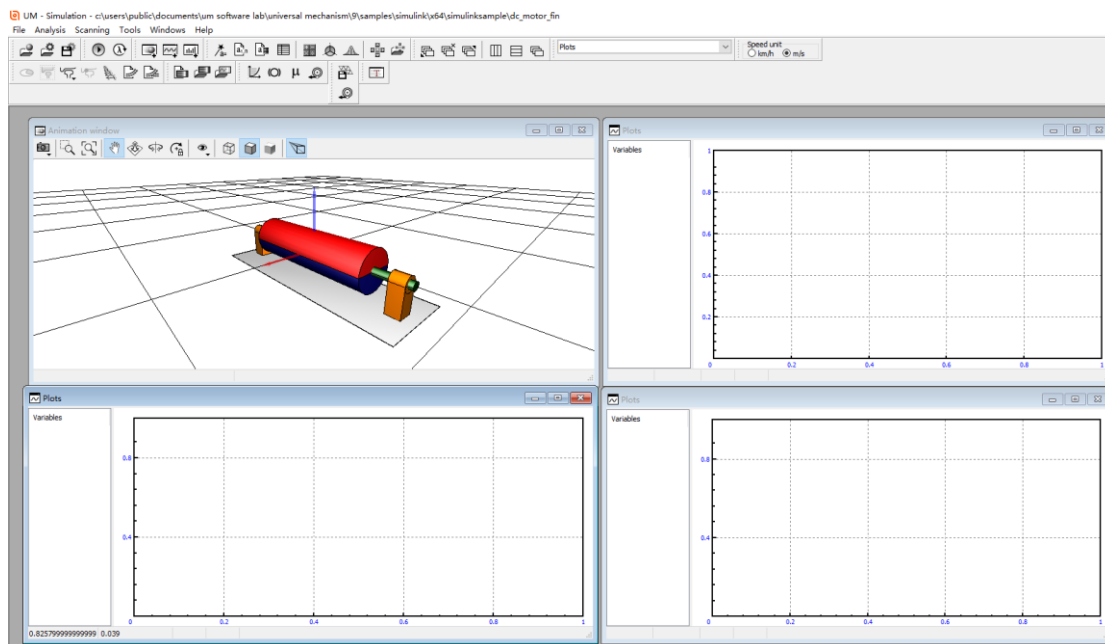




图 3.22 仿真界面窗口布局

3. 选择菜单 **Tools | Wizard of variables**，打开变量向导。
4. 点击变量向导的 **Identifier** 页面，在左侧列表选中 **Ma**，然后点击按钮  创建变量 **Ma**。
5. 点击变量向导的 **Joint force** 页面，在左侧列表选中 **jShaft**，然后设置类型为 **Torque**，分量为 **Y**，选择 **Acts on** 物体为 **body2:Shaft**，点击按钮  创建变量 **jAFy(jShaft)**。
6. 将变量 **Ma** 和 **jAFy(jShaft)** 拖入图 3.22 所示左下方的绘图窗口。
7. 用同样的方法创建角速度变量（**Angular Variable** 页面，**Type of variable** 为 **Ang.velocity**，**Component** 为 **Y**），并拖入右上方的绘图窗口。
8. 用同样的方法创建电流和电压变量（**Identifier** 页面，**I** 和 **U** 参数），并拖入右下方的绘图窗口。
9. 关闭变量向导。

运动仿真

1. 选择菜单 **Analysis | Simulation**，打开仿真控制界面。
2. 点击 **Solver** 页面，选择求解器 **BDF**，仿真时间 **10s**，动画和数据步长 **0.02s**，容差 **0.001**。
3. 点击 **Integration** 开始仿真计算。
4. 仿真过程如图 3.23 所示。

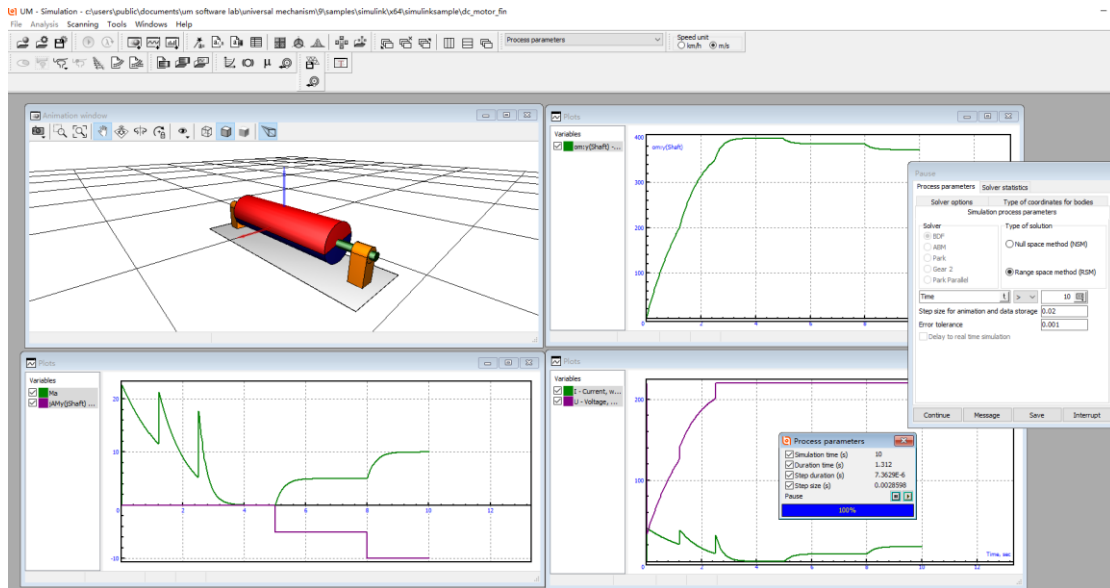


图 3.23 仿真过程

4. 使用 CoSimulation 工具

下面我们使用 **UM Control/CoSimulation** 工具来进行联合仿真，需要将 **UM** 机械系统模型作为 **S** 函数导入到 **Matlab/Simulink**。

4.1 工作流程

从 **UM** 导出机械系统模型到 **Matlab/Simulink** 进行联合仿真的流程如下：

- 在 **Matlab/Simulink** 里搭建控制框图；
- 添加一个 **S-function** 组件到 **Matlab/Simulink** 控制系统；
- 在 **UM Input** 程序里创建机械系统模型；
- 在 **UM Simulation** 程序里加载机械系统模型，设置好连接，生成 **m-file**；
- 在 **Matlab/Simulink** 里进行动力学仿真。

将 **UM** 模型考虑为一个具有若干个输入和输出信号的黑盒子。输入信号为 **UM Input** 程序里创建的参数符号，通常与力/力矩关联；输出信号为 **UM Simulation** 程序里变量向导 (**Wizard of variables**) 创建的变量，如速度、加速度等。

4.2 倒立摆

UM 软件自带的倒立摆模型（已准备好 `.cosim` 和 `.m` 文件）位于本地目录 `{UM Data} \SAMPLES\cosimulation\inv_pend_cosim`。在开始学习本课程之前，请先确认这个模型是否存在。如果没有找到，可以从 UM 软件官方网站下载：www.umlabor.ru/download/90/inv_pend_cosim.zip。

我们将直接使用这个模型，这里就不再详细介绍其建模过程，而着重讲解有关连接 **Matlab/Simulink** 模型的方法。

4.2.1 准备 Matlab/Simulink 模型

本例用到的 **Matlab/Simulink** 控制系统模型与图 3.1 的模型非常相似，如图 4.1 所示。不同之处在于，这里用到了一个 **S-Function** 模块，这个模块可以在 **Simulink** 的 **User-Defined Functions** 库找到，缺省名称为 **System**。

通过 **S-Function** 可以将 **UM** 模型和 **Simulink** 模型连接起来。如图 4.1 所示，**S-Function** 有一个输入信号（力）和一个输出信号（角度）。

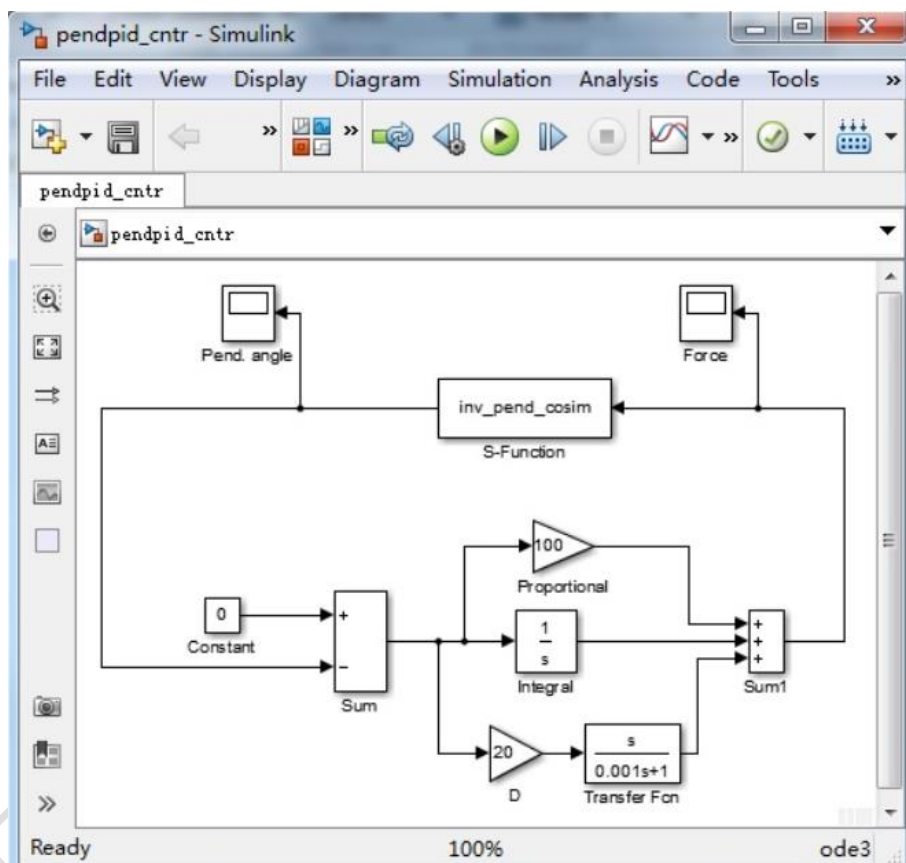


图 4.1 包含 S-Function 的控制系统

图 4.1 对应的 **Matlab/Simulink** 模型位于本地目录 {UM Data}\SAMPLES\cosimulation\inv_pend_cosim\pendpid_ctr。

接下来，我们先从 **UM** 输出模型，再到 **Simulink** 里进行仿真计算。

4.2.2 输出 UM 模型

下面，我们在 **UM** 模型里定义与 **S-Function** 相关的输入和输出信号，保存相关设置，并生成专门的 **m** 文件。

加载 UM 模型

1. 运行 **UM Simulation**。
2. 打开模型 {UM Data} \SAMPLES\cosimulation\inv_pend_cosim。

设置初始条件

首先，我们需要将倒立摆初始转动一定角度，使之偏离理想的平衡位置。

1. 选择菜单 **Analysis | Simulation**，打开仿真控制界面。
2. 点击 **Initial conditions** 页面。
3. 设置 **Coordinate/1.2** 为 **0.3**，如图 4.2 所示。

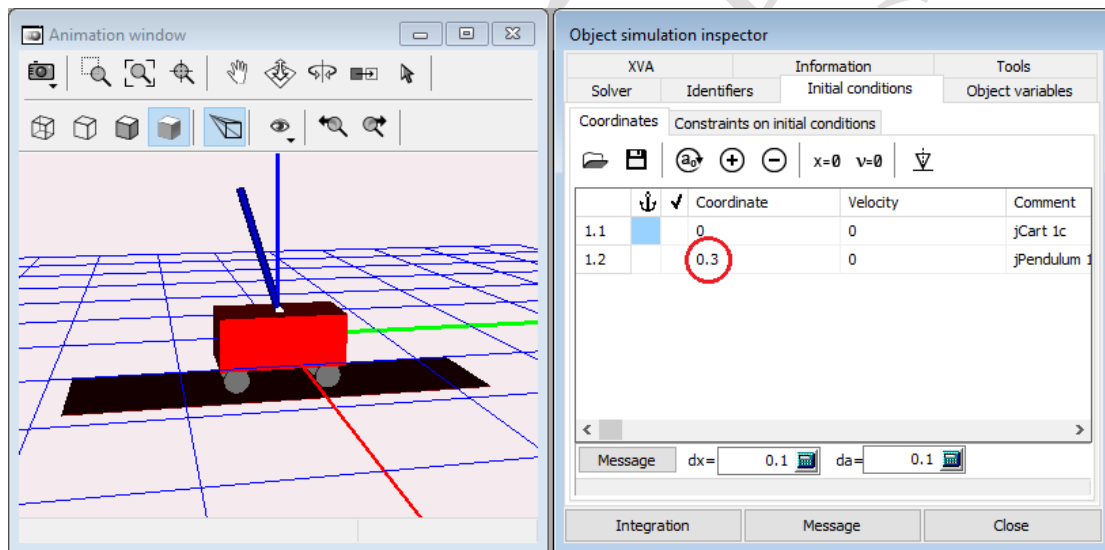


图 4.2 设置初始条件

输出 UM 模型

1. 选择菜单 **Tools | Wizard of export**，弹出 **UM** 输出向导窗口。

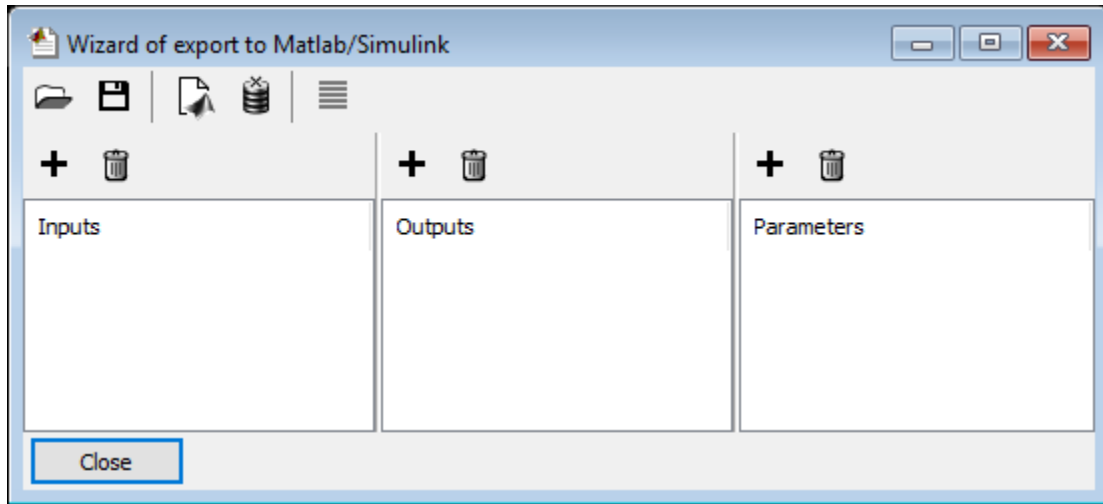


图 4.3 UM 输出向导窗口

现在我们需要创建两个变量：一个是作用于小车的控制力变量，作为输入信号；另一个是倒立摆的竖向摆角，作为输出信号。

定义输入信号

1. 在图 4.3 界面 **Inputs** 框点击按钮 **+**，弹出模型参数窗口。
2. 从下拉菜单选择 **force** 参数。
3. 点击 **OK**，如图 4.4 所示。

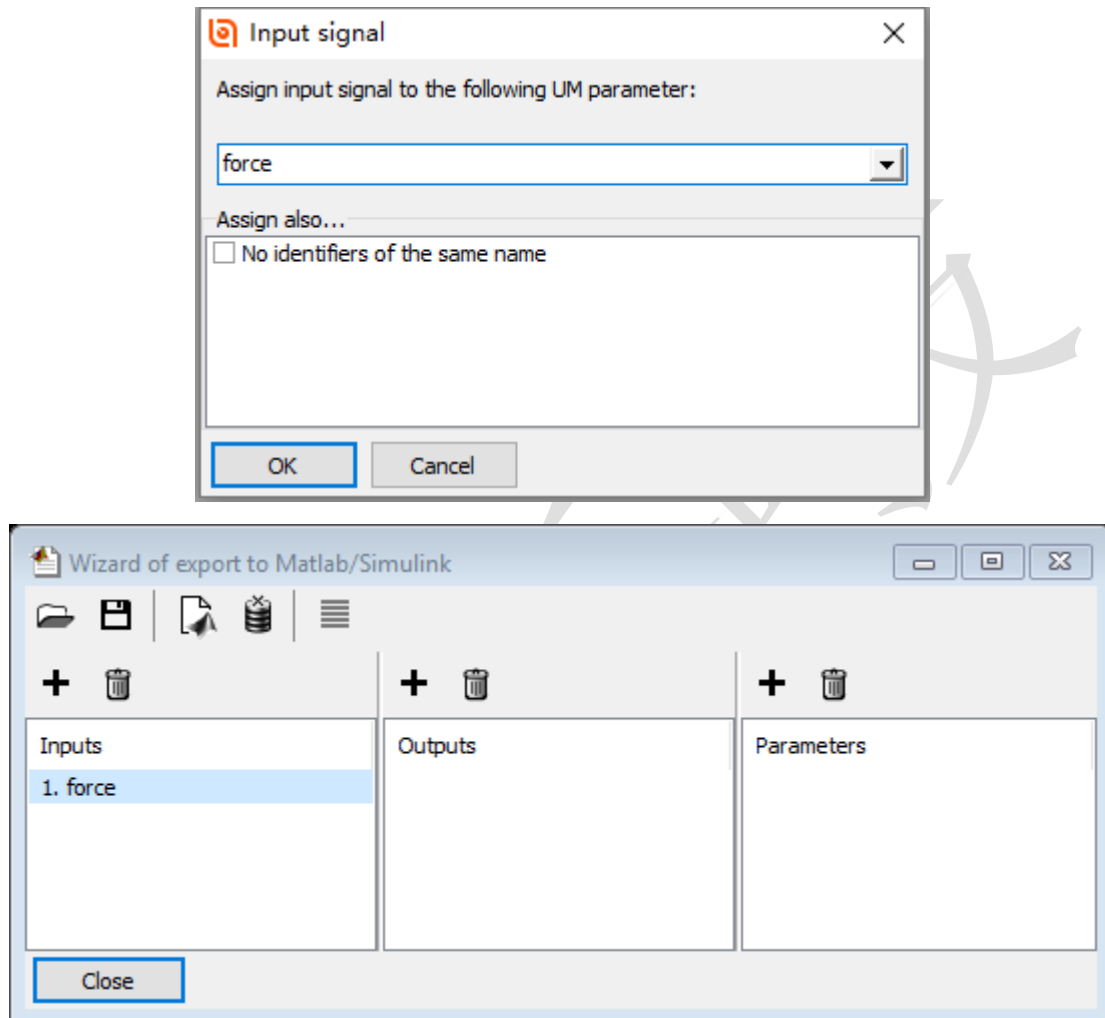



图 4.4 定义输入信号

定义输出信号

1. 在图 4.3 界面 **Outputs** 框点击按钮 **+**，添加一个输出信号。
2. 打开变量向导。
3. 点击 **Angular variables** 页面。
4. 在左侧列表选中 **Pendulum**，取消选择 **Use orientation at zero coordinates**，设置 **Type of variable** 为 **Rot.vector**，**Component** 为 **X**。
5. 点击按钮 ，创建角度变量 **ang:x(Pendulum)**。

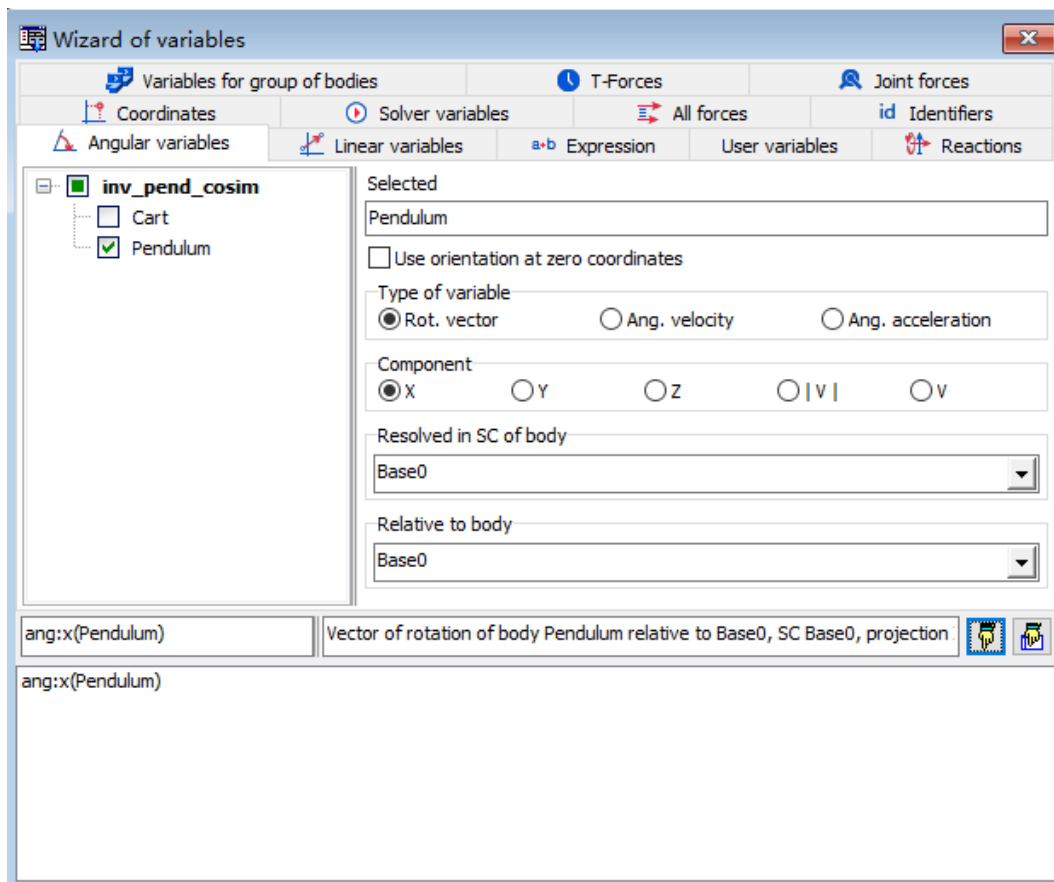


图 4.5 变量向导

- 将上一步创建的变量 **ang:x(Pendulum)** 从变量向导拖入输出向导的 **Output1** 处，作为输出信号，如图 4.6 所示。

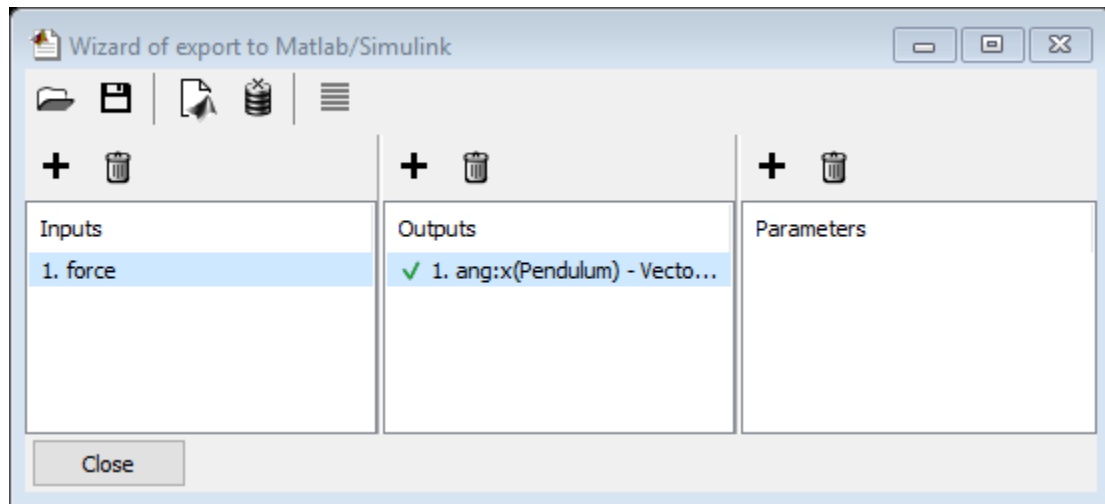


图 4.6

至此，UM 模型输出前的准备工作已经全部完成，下面来生成 **.m** 文件和 **.cosim** 文件。

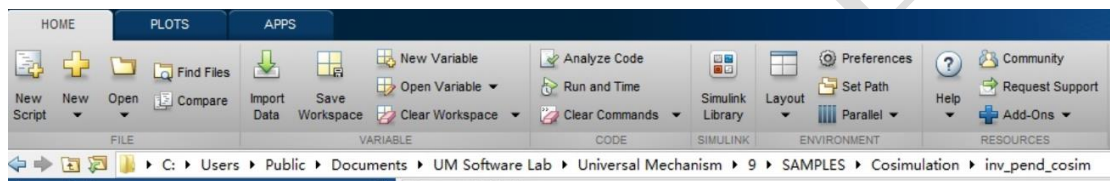
生成 m 文件

- 在图 4.3 界面点击按钮 ，弹出警告信息，提示参数列表为空，点击 **Yes**，忽略即可。
- 弹出保存窗口，设置名称 **inv_pend_cosim**，点击 **Save**。这样就生成了 **inv_pend_cosim.cosim** 文件和 **inv_pend_cosim.m** 文件。在 **Matlab/Simulink** 环境中执行仿真时，通过 **cosim** 文件与 **UM** 进行数据交互。

4.2.3 连接 UM 模型和 Matlab/Simulink 模型

在前面的操作过程中，生成的 **m** 文件缺省保存于 **UM** 模型目录。为了仿真顺利进行，需要将其复制到 **Matlab/Simulink** 模型所在目录。只是由于本例的 **Matlab/Simulink** 模型已经位于 **UM** 模型目录，因此这里无需操作。

1. 运行 **Matlab/Simulink**。
2. 设置当前工作目录为 {UM Data}\SAMPLES\cosimulation
inv_pend_cosim。



3. 加载图 4.1 所示的控制系统模型 **pendpid_ctr**。
4. 双击 **S-Function**，弹出参数设置窗口。
5. 设置名称为 **inv_pend_cosim**，即 **m** 文件的名称，如图 4.7 所示。
6. 点击 **OK**。

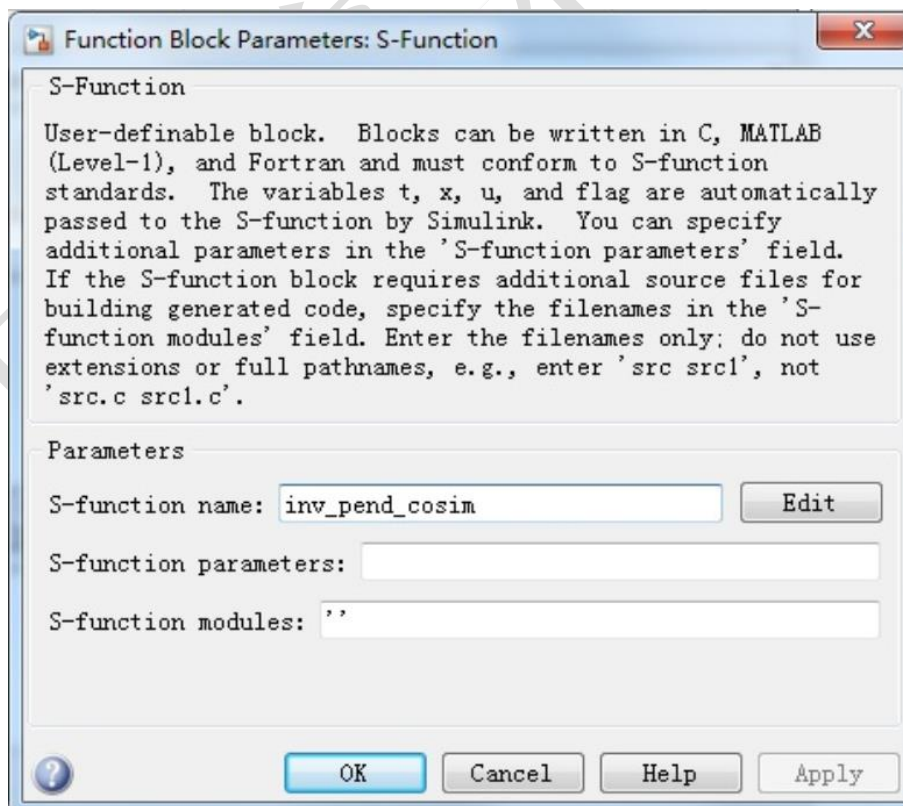
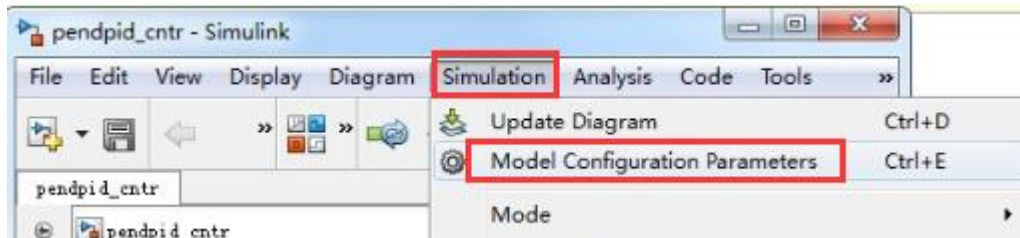


图 4.7 设置 S-Function 名称

4.2.4 运动仿真

设置仿真参数

1. 选择菜单 **Simulation | Configuration Parameters**。



2. 点击 **Solver** 页面。
3. 设置 **Start time** 为 **0.0**，**Stop time** 为 **0.3**，选择 **Type** 为 **Fixed-step**，**Solver** 为 **ode3 (Bogacki-Shampine)**，如图 4.8 所示。

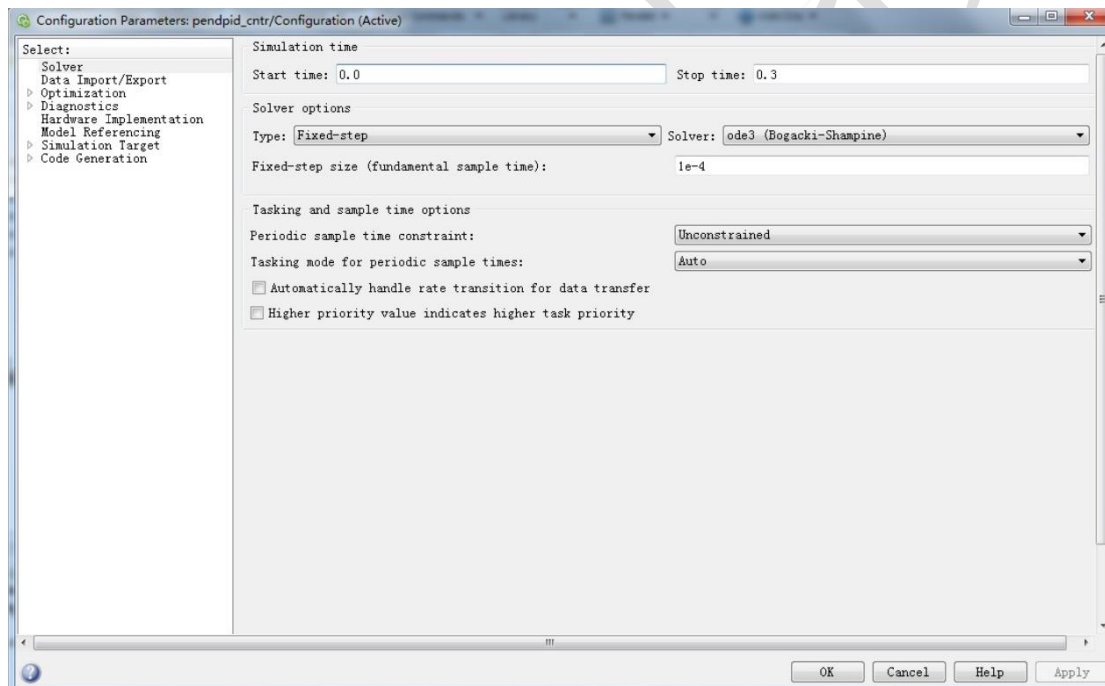


图 4.8 仿真参数

4. 点击 **OK**。

运动仿真

1. 双击打开所有的 **Scope** 窗口。
2. 运行仿真。
3. 仿真结果如图 4.9 和图 4.10 所示。

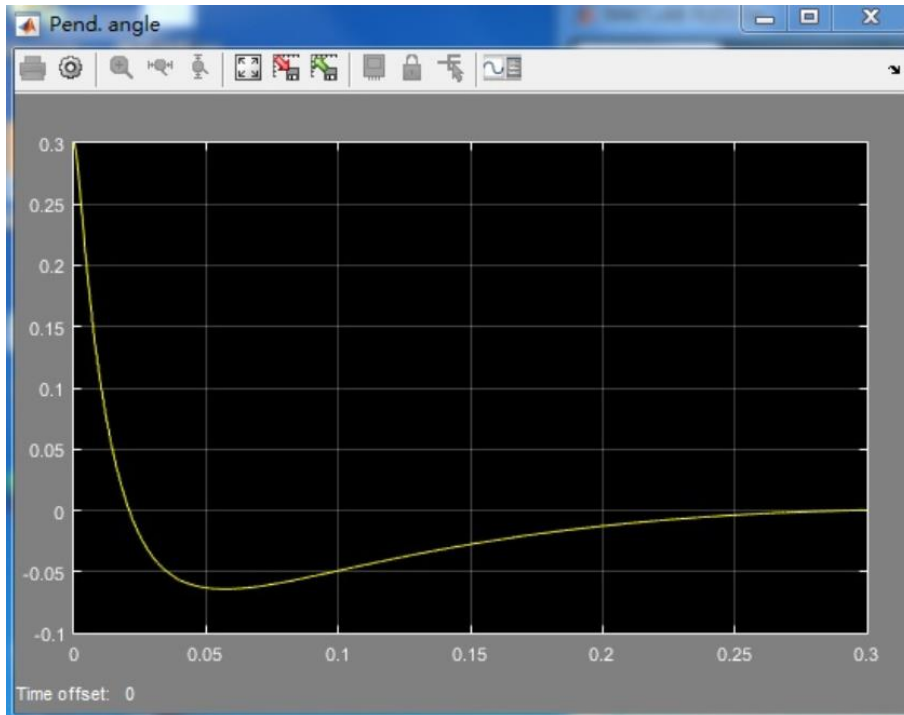


图 4.9 倒立摆竖向摆角

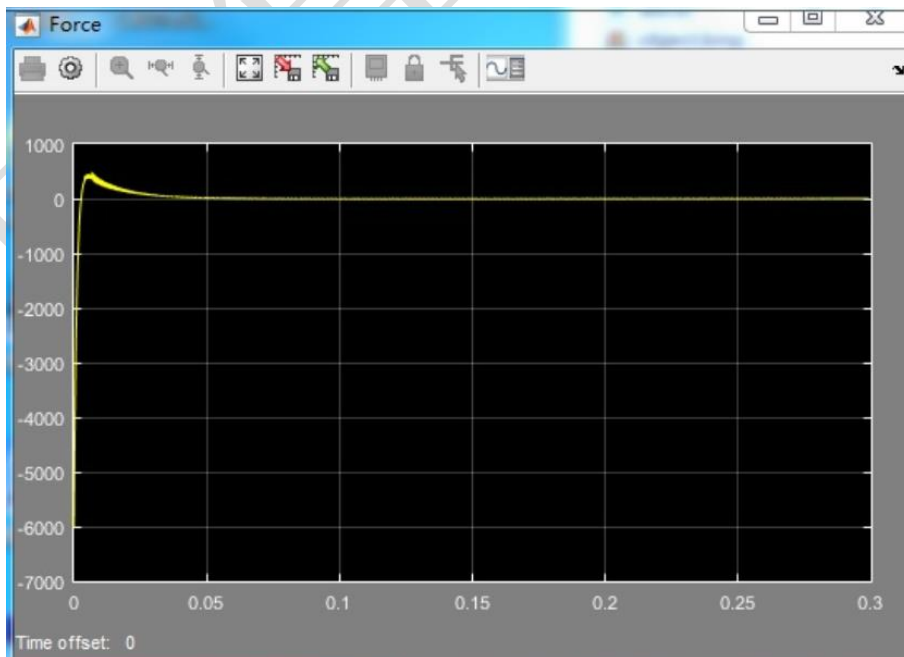


图 4.10 作用在小车上的控制力

4.3 直流电机

UM 软件自带的直流电机模型（已准备好`.cosim` 和`.m` 文件）位于本地目录 `{UM Data} \SAMPLES\cosimulation\dcmotor_cosim`。在开始学习本课程之前，请先确认这个模型是否存在。如果没有找到，可以从 UM 软件官方网站下载：www.umlab.ru/download/90/dcmotor_cosim.zip。

4.3.1 准备 Matlab/Simulink 模型

本例用到的 Matlab/Simulink 控制系统模型与图 3.16 的模型非常相似，如图 4.11 所示。不同之处在于，这里用到了一个 S-Function 模块，这个模块可以在 Simulink 的 User-Defined Functions 库找到，缺省名称为 System。控制系统模型位于 `{UM Data} \SAMPLES\cosimulation\dcmotor_cosim\dcmotor_cntr`。

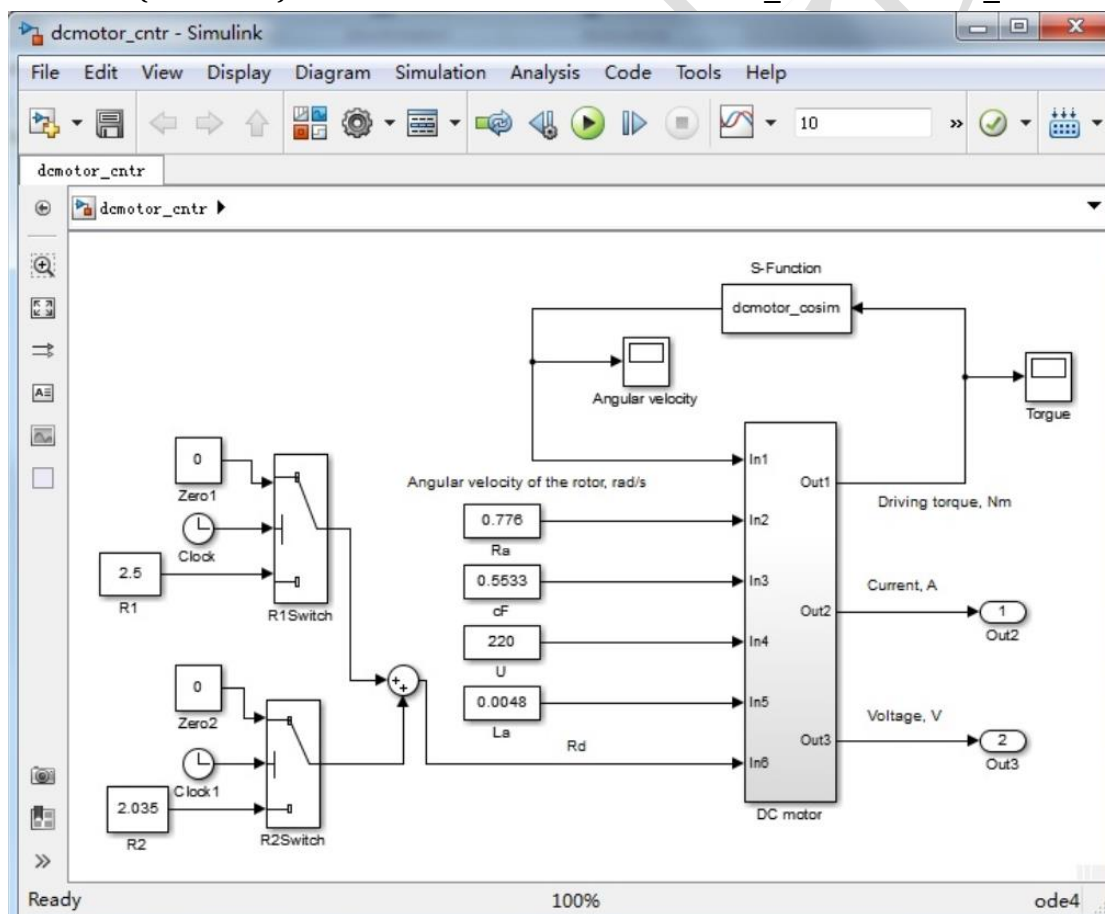


图 4.11

4.3.2 输出 UM 模型

加载 UM 模型

1. 运行 **UM Simulation**。
2. 打开模型 {UM Data} \SAMPLES\cosimulation\dcmotor_cosim。

输出 UM 模型

1. 选择菜单 **Tools | Wizard of export**，弹出输出向导窗口。
现在我们需要创建两个变量：一个是作用于传动轴的控制力矩，作为输入信号；另一个是传动轴的角速度，作为输出信号。

定义输入信号

1. 在 **Inputs** 框点击按钮 **+**，弹出模型参数窗口。
2. 从下拉菜单选择 **Ma** 参数，点击 **OK**，如图 4.12 所示。

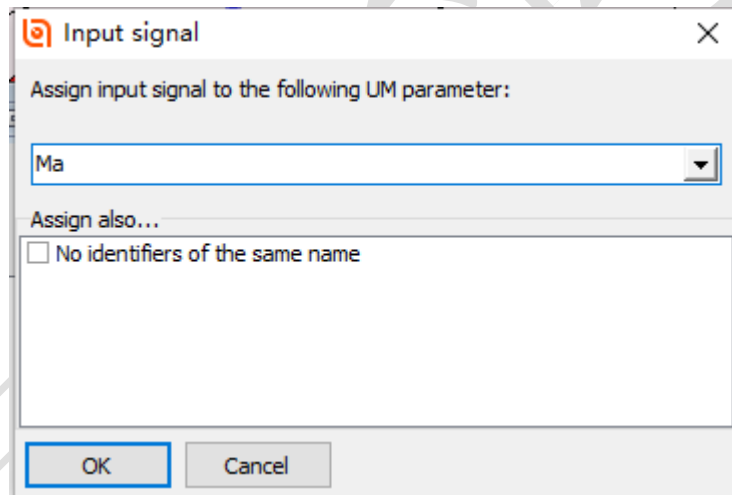



图 4.12 定义输入信号

定义输出信号

1. 在 **Outputs** 框点击按钮 **+**，添加一个输出信号。
2. 打开变量向导。
3. 点击 **Angular variables** 页面。
4. 在左侧列表选中 **Shaft**，设置 **Type of variable** 为 **Ang.velocity**，**Component** 为 **Y**。点击按钮 ，创建变量 **om:y(Shaft)**，如图 4.13 所示。

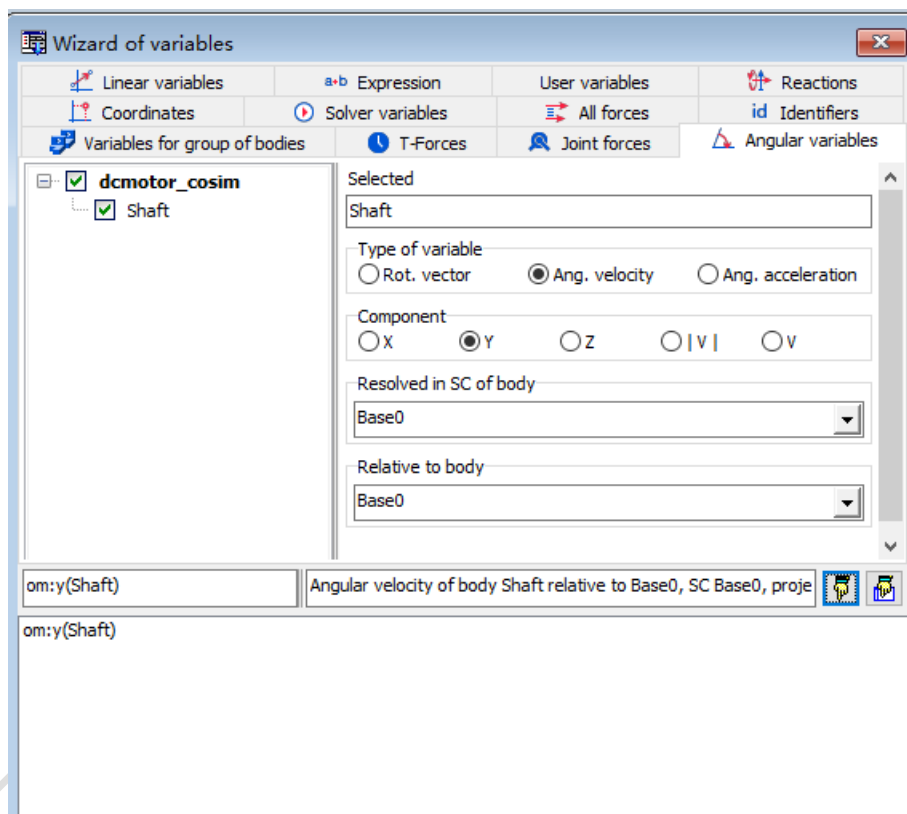


图 4.13 变量向导

5. 将变量 **om:y(Shaft)** 从变量向导拖入输出向导的 **Output1** 处，作为输出信号，如图 4.14 所示。

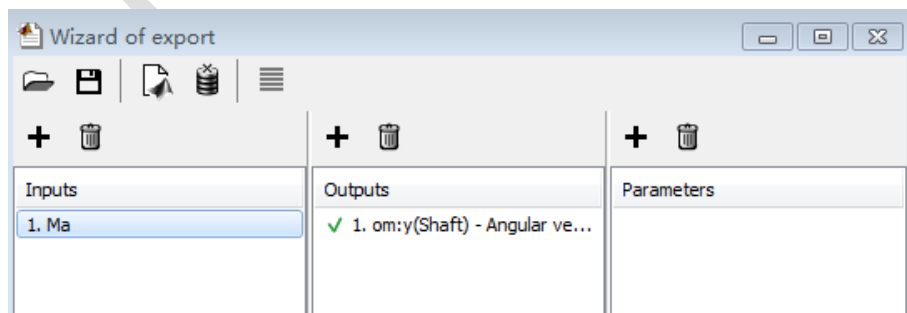



图 4.14

生成 m 文件

1. 在点击按钮，弹出警告信息，提示参数列表为空，点击 **Yes**，忽略即可。
2. 弹出保存窗口，设置名称 **dcmotor_cosim**，点击 **Save**。

4.3.3 连接 UM 模型和 Matlab/Simulink 模型

1. 运行 **Matlab/Simulink**。
2. 加载图 4.11 所示的控制系统模型 **dcmotor_cosim**。
3. 双击 **S-Function**，弹出参数设置窗口。
4. 设置名称为 **dcmotor_cosim**，即 **m** 文件的名称，如图 4.15 所示。
5. 点击 **OK**。

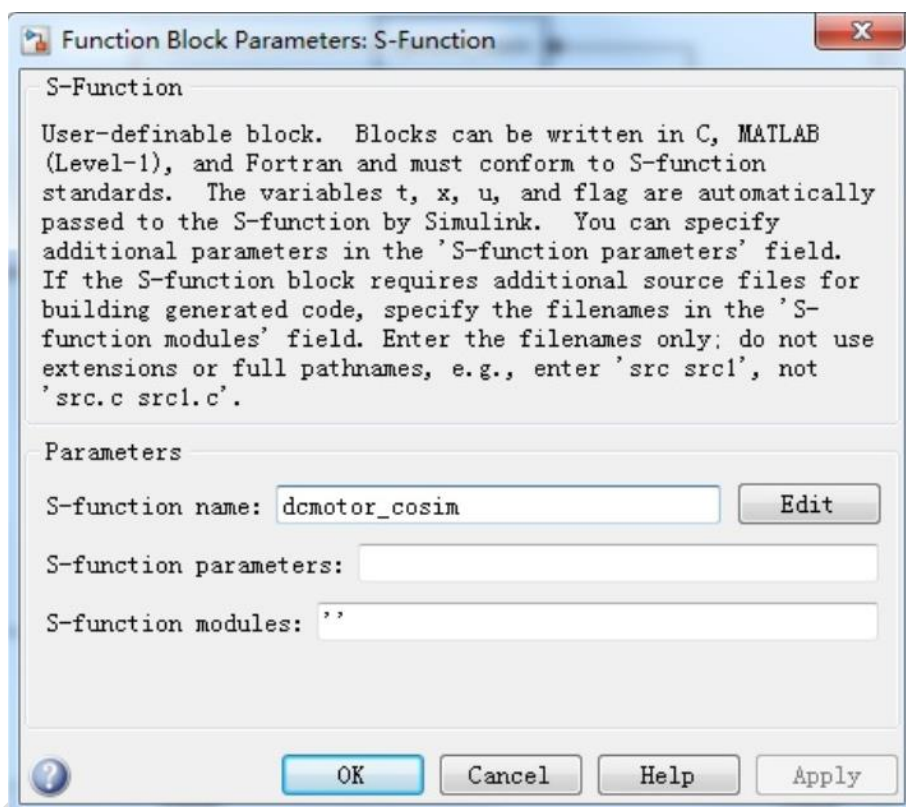


图 4.15 设置 S-Function 名称

4.3.4 运动仿真

运动仿真

1. 双击打开所有的 **Scope** 窗口。
2. 运行仿真。
3. 仿真结果如图 4.16 和图 4.17 所示。



图 4.16 角速度

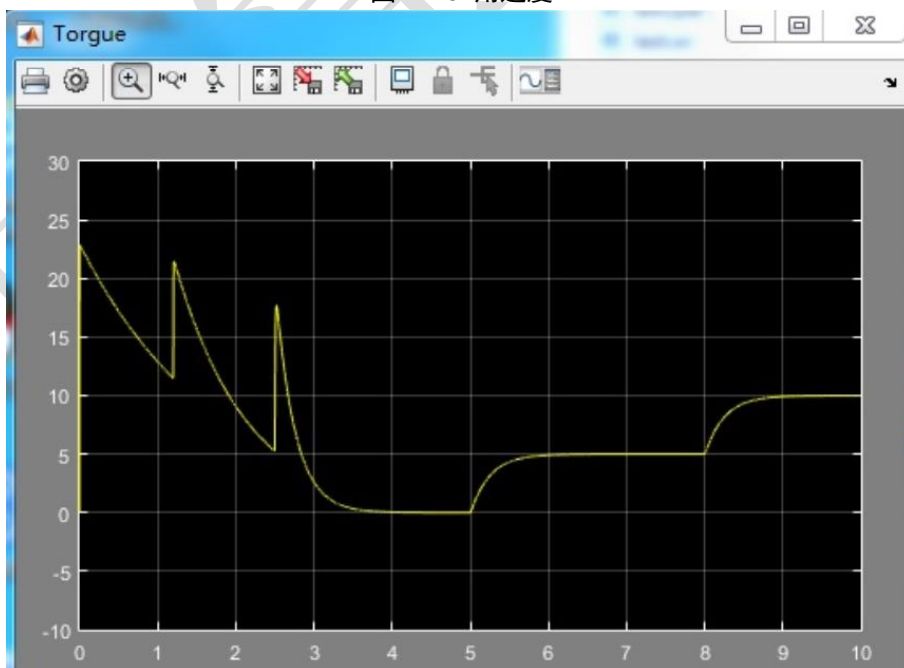


图 4.17 驱动力矩