第七章 Simulink 基础

7.1 Simulink 简介

Simulink 是一个用来对动态系统进行建模、仿真和分析的软件包，它支持连续、离散及两者混合的线性和非线性系统，也支持具有多种采样频率的系统。在 Simulink 环境中，利用鼠标就可以在模型窗口中直观地“画”出系统模型，然后直接进行仿真。它为用户提供了方框图进行建模的图形接口，采用这种结构画模型就像你用手和纸来画一样容易。它与传统的仿真软件包微分方程和差分方程建模相比，具有更直观、方便、灵活的优点。Simulink 包含有 Sinks（输出方式）、Source（输入源）、Linear（线性环节）、Nonlinear（非线性环节）、Connections（连接与接口）和 Extra（其他环节）等子模型库，而且每个子模型库中包含有相应的功能模块，用户也可以定制和创建自己的模块。

用 Simulink 创建的模型可以具有递阶结构，因此用户可以采用从上到下或从下到上的结构创建模型。用户可以从最高级开始观看模型，然后用鼠标双击其中的子系统模块，来查看其下一级的内容，以此类推，从而可以看到整个模型的细节，帮助用户理解模型的结构和各模块之间的相互关系。在定义完一个模型后，用户可以通过 Simulink 的菜单或 MATLAB 的命令窗口键入命令来对它进行仿真。菜单方式对于交互工作非常方便，而命令行方式对于运行一大类仿真非常有用。采用 Scope 模块和其他的画图模块，在仿真进行的同时，就可观看到仿真结果。除此之外，用户还可以在改变参数后迅速观看系统中发生的变化情况。仿真的结果还可以存放到 MATLAB 的工作空间里做事后处理。

模型分析工具包括线性化和平衡点分析工具、MATLAB 的许多基本工具箱及 MATLAB 的应用工具箱。由于 MATLAB 和 Simulink 是集成在一起的，因此用户可以在这两种环境下对自己的模型进行仿真、分析和修改。

Simulink 具有非常高的开放性，提倡将模型通过框图表示出来，或者将已有的模型添加组合到一起，或者将自己创建的模块添加到模型当中。Simulink 具有较高的交互性，允许随意修改模块参数，并且可以直接无缝地使用 MATLAB 的所有分析工具。对最后得到的结果可进行分析，并能够将结果可视化显示。

Simulink 非常实用，应用领域很广，可使用的领域包括航空航天、电子、力学、数学、通信、影视和控制等。世界各地的工程师都在利用它来对实际问题进行建模、分析和解决。

7.2 Simulink 的基本操作

7.2.1 Simulink 的运行

运行 Simulink 有三种方式：

- 在 MATLAB 的命令窗口直接键入“Simulink”并回车；
- 单击 MATLAB 工具条上的 Simulink 图标；
- 在 MATLAB 菜单上选 File→New→Model。

运行后会显示图 7.2.1 所示的 Simulink 模块库浏览器，单击工具条左边建立新模型的快捷方式，则显示如图 7.2.2 所示的新建模型窗口，在模型窗口中用户便可通过选择模块库中的仿真模块，建立自己的仿真模型，并进行动态仿真。

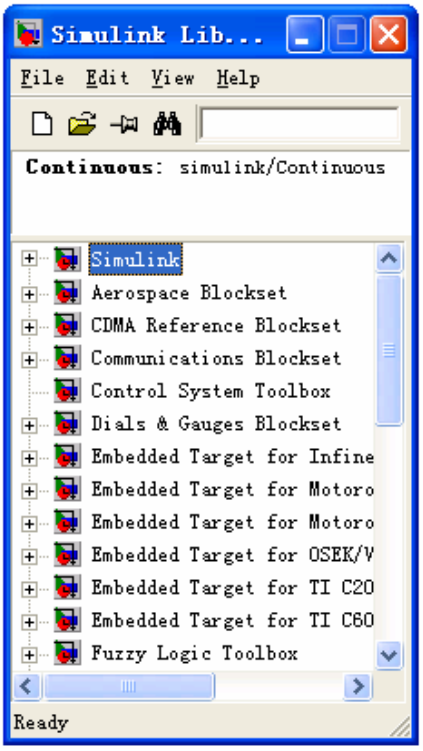

图 7.2.1 Simulink 模块库浏览器

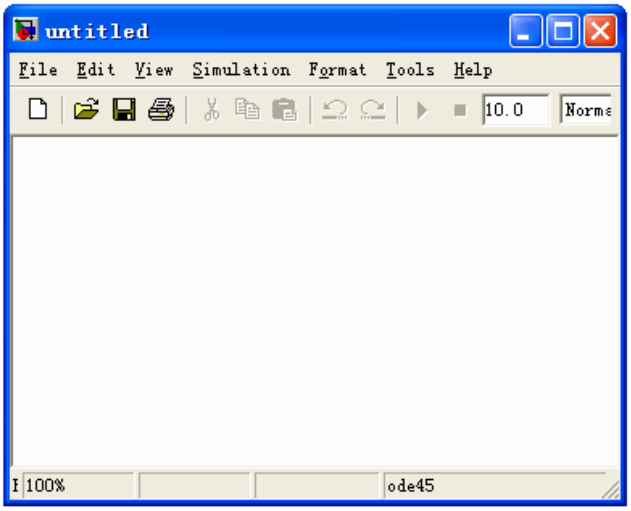

图 7.2.2 新建模型窗口

7.2.2 常用的标准模块

附录 C 以表格的形式给出 Simulink 几个基本模块库中的模块功能简介，表格中的模块名和模块库中的模块图标下的名称一致。打开模块库（图标）窗口的方法非常简单，以连续系统模块库（continuous）为例，在 Simulink 模块库浏览窗口中选中 Simulink，然后单击 Simulink 旁边的小加号或者双击鼠标左键，这时就会出现如图 7.2.3 所示 Simulink 基本库窗口，并选择 Continuous 模块库的图标双击即可进入如图 7.2.4 所示的连续系统模块库，可选择相应的模块图标拖至编辑窗口即可。

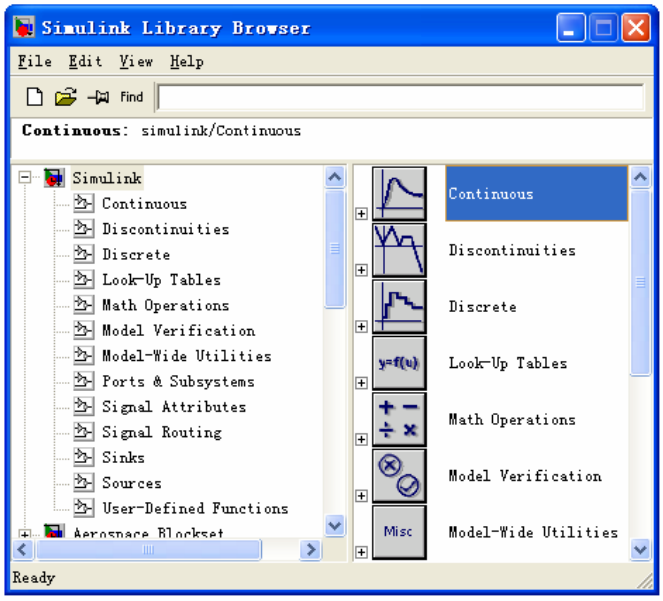

图 7.2.3 Simulink 模型库窗口

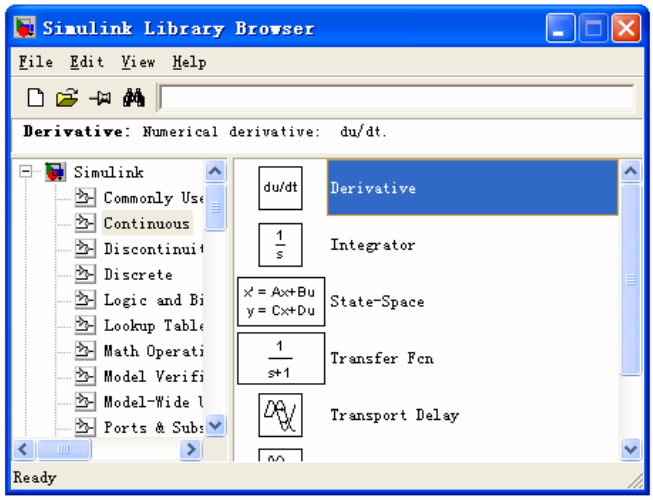

图 7.2.4 continuous 模块库

7.2.3 模块的操作

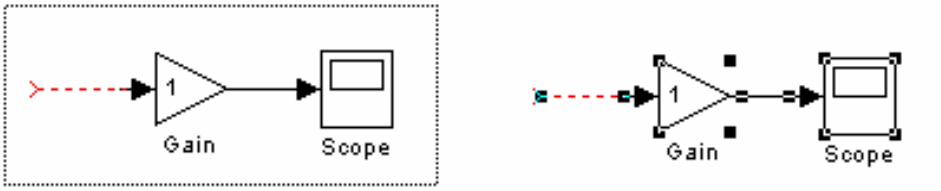

图 7.2.5 选取模块

1、模块的选取

当选取单个模块时，只要用鼠标在模块上单击即可，此时模块的角上出现黑色小方块。选取多个模块时，选取拖拽鼠标的方式把要选择的模块全部包围即可，若所有被选取的模块都出现小黑方块，则表示模块被选中，如图 7.2.5 所示。

2、模块的复制、剪切、删除、移动

应用【Edit】|【copy】/【cut】/【paste】/【clear】可对选取的模块进行复制，剪切，粘贴，

删除等操作，如果要在同一窗口移动模块，则在模块选中的基础上，用鼠标进行拖拽并放在合适的位置。

3、模块的连接

(1) 连接两个模块：从一个模块的输出端连到另一个模块的输入端。如果两个模块不在同一水平线上，连线是折线，若用斜线表示则需在连接时按住【Shift】。

(2) 在连线之间插入：把模块用鼠标拖到连线上，然后释放鼠标即可。

(3) 连线的分支：当我们需要把一个信号输送给不同的模块时，连线要采用分支结构，其操作步骤是：先连好一条线，把鼠标移到支线的起点，并按下【Ctrl】，再将鼠标拖至目标模块的输入端即可。

4、模块参数的设置

Simulink 中几乎所有模块的参数（Parameters）都允许用户进行设置，只要双击要设置的模块或在模块上按鼠标右键并在弹出的菜单中选择【Block Parameters】就会显示参数设置对话框。

例7.2.1 已知单位负反馈二阶系统的开环传递函数为

$$G(s)=\frac{10}{s^2+4.47s}$$

试绘制单位阶跃响应的Simulink结构图。

解：1、利用Simulink的Library窗口中的【File】|【New】，打开一个新的工作空间；

2、分别从信号源库（Sourse）、输出方式库(Sink)、数学运算库(Math)、连续系统库（Continuous）中，用鼠标把阶跃信号发生器（Step）、示波器（Scope）、传递函数（Transfer Fcn）、相加器（Sum）四个标准功能模块选中，并将其拖至工作平台；

3、按要求先将前向通道连接好，然后把相加器（Sum）的另一个端口与传递函数和示波器间的线段相连，形成闭环反馈；

4、双击阶跃信号发生器，打开其属性设置对话框，并将其设置为单位阶跃信号，如图7.2.6所示，同理，将相加器设置为“+-”，使传递函数的Numerator设置为“［10］”，Denominator设置为“[1 4.47 0]”；

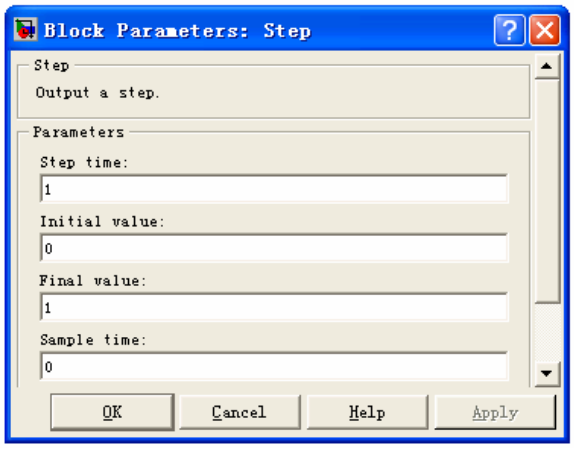

图 7.2.6 模块参数设置对话框

5、绘制成功后，如图7.2.7所示，并命名后存盘。

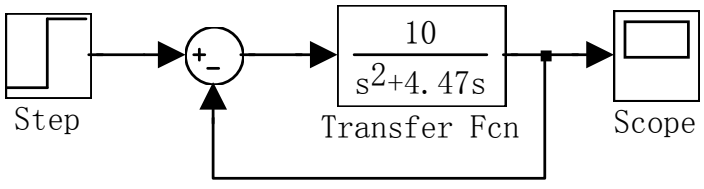

图7.2.7二阶系统Simulink结构图

5、模块外形的调整

(1) 改变模块的大小：选定模块，用鼠标点住其周围的四个黑方块中的任意一个拖动，这时会出现一个虚线的矩形表示新模块的位置，到需要的位置后释放鼠标即可。

(2) 调整模块的方向：选定模块，选择菜单【Formt】|【Rotate Block 】使模块旋转 90°，【Flip Block】使模块旋转 180°。

(3) 给模块加阴影：选定模块，选择菜单【Formt】|【Show Drop Shadow】使模块产生阴影效果。

6、模块名的处理

(1) 模块名的显示与消隐：选定模块，选择菜单【Format】|【Filp Name】使模块名被隐藏，同时【Show Name】会使隐藏的模块名显示出来。

(2) 修改模块名：用鼠标左键单击模块名的区域，使光标处于编辑状态，此时便可对模块名进行任意的修改。同时选定模块，选择菜单 【Format】|【Font】可弹出字体对话框，用户可对模块名和模块图标中的字体进行设置。

例 7.2.2 将图 7.2.7 所示的结构图进行模块处理。

解：

1. 对模块名进行修改，如单击传递函数模块标题“Transfer Fcn”，将其原字符删除，并输入汉字“传递函数”，同理将其他模块也改为汉字标题；
2. 将相加器的标题移至其顶部；
3. 选中“传递函数”模块，并选择菜单【Formt】|【Show Drop Shadow】并将其设置为阴影；
4. 将模块全部选中，选择菜单【Format】|【Font】通过字体对话框将所有字体设置为“宋体”，如图 7.2.8 所示。

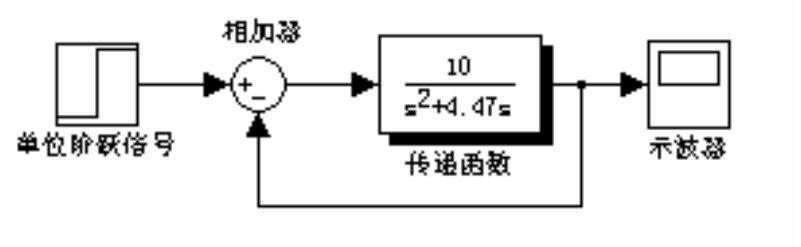

图 7.2.8 二阶系统模型

7.3 系统仿真及参数设置

在 Simulink 中建立起系统模型框图后，运行菜单【Simulation】|【Start】就可以用 Simulink 对模型进行动态仿真。一般在仿真运行前需要对仿真参数进行设置，运行菜单【Simulation】|【Parameters】完成设置，如图 7.3.1 所示。

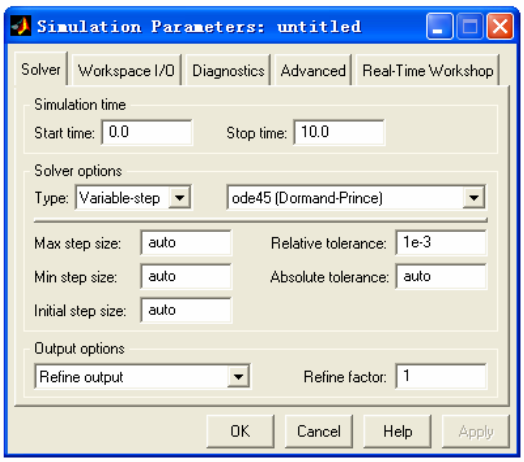

图 7.3.1 仿真参数设置对话框

7.3.1 算法设置

在 Solver 里需要设置仿真起始和终止时间、选择合适的解法（Solver）并指定参数、设置一些输出选择。

1　设置起始时间和终止时间（Simulation time）

【Simulation】|【Start time】设置起始时间，而【Stop time】设置终止时间，单位为“秒”。

2　算法设置(Solver option)

(1) 算法类型设置

仿真的主要过程一般是求解常微分方程组，【Solver option】|【Type】用来选择仿真算法的类型是变化的还是固定的。

变步长解法可以在仿真过程中根据要求调整运算步长，在采用变步长解法时，应该先指定一个容许误差限（【Relative tolerance】或【Absolute tolerance】），使得当误差超过误差限时自动修正仿真步长，【Max step size】用于设置最大步长，在缺省情况下为“auto”，并按下式计算最大步长：

最大步长=（终止时间-起始时间）/50。

(2) 仿真算法设置

离散模型：对变步长和定步长解法均采用 discrete(no continuous state)。

连续模型：可采用变步长和定步长解法。

变步长解法有：ode45、ode23、ode113、ode15s、ode23s、ode23t，ode23st

ode45：四阶/五阶 Runge-Kutta 算法，属单步解法；

ode23：二阶/三阶 Runge-Kutta 算法，属单步解法；

ode113：可变阶次的 Adams-Bashforth-Moulton PECE 算法，属于多步解法；

ode15s：可变阶次的数值微分公式算法，属于多步解法；

ode23s：基于修正的 Rosenbrock 公式，属单步解法。

定步长解法有：ode4、ode5、ode3、ode2、ode1

ode5：定步长的 ode45 解法；

ode4：四阶 Runge-Kutta 算法；

ode3：定步长 ode23 算法；

ode2：Henu 方法，即改进的欧拉法。

ode1：欧拉法。

3　设置输出选项

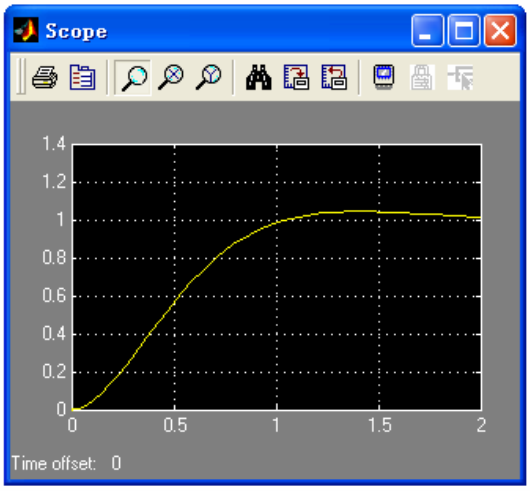

图 7.3.2 二阶系统的仿真曲线

对同样的信号，选择不同的输出选项，则得到输出设备上的信号是不完全一样的。要根据

需要选择合适的输出选项以达到满意的输出效果。

对于例 7.2.2，首先运行菜单【Simulation】|【Parameters】，进行系统的仿真参数设置，如仿真时间为 2 秒，仿真算法选择定步长的四阶龙格—库塔法，然后运行菜单【Simulation】|【Start】进行系统仿真，最后双击示波器，得到系统的仿真曲线如图 7.3.2 所示。

7.3.2 工作空间设置

工作空间设置（**Workspace I/O**）窗口如图 7.3.3 所示，可以设置 Simulink 和当前工作空间的数据输入、输出。通过设置，可以从工作空间输入数据、初始化状态模块，也可以把仿真结果、状态变量、时间数据保存到当前工作空间。

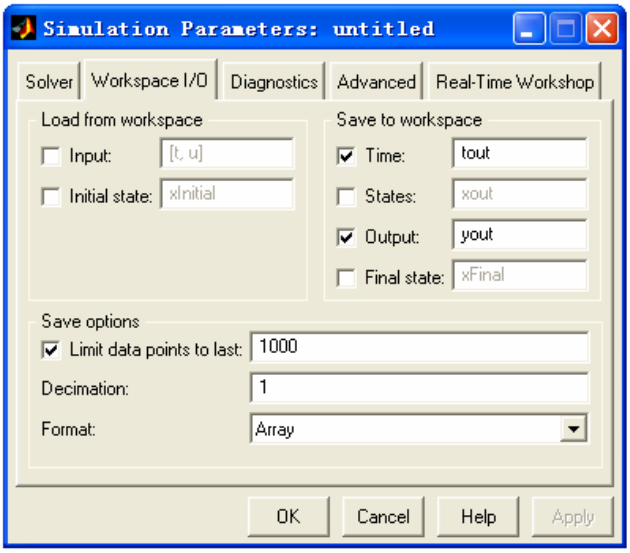

图 7.3.3 设置 Workspace I/O 窗口

1、从工作空间读入数据（Load from workspace ）

Simulink 通过设置模型的输入端口，实现在仿真过程中从工作空间读入数据，常用的输入端口模块为信号与系统模块库（Signals & Systems）中的 In1 模块，其参数设置如图 7.3.4 所示。

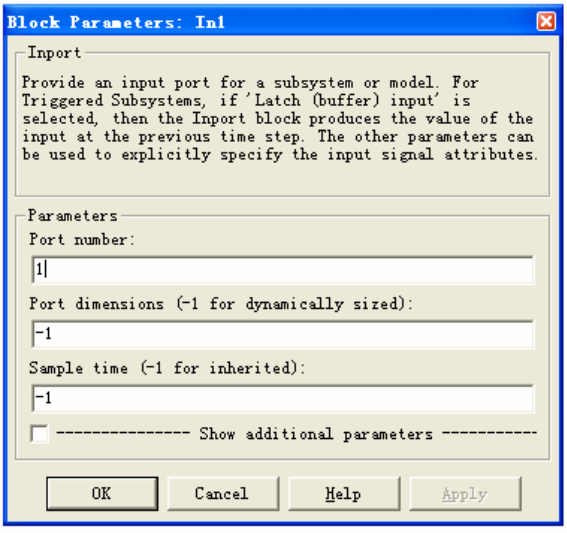

图 7.3.4 输入参数设置对话框

设置的方法是选中 Input 前的复选框，并在后面的编辑框键入输入数据的变量名，并可以用命令窗口或 M 文件编辑器输入数据。Simulink 根据输入端口参数中设置的采样时间读取输入数据。

2、保存数据到工作空间(Save to workspace)

可以选择保存的选项有：时间、端口输出、状态、最终状态。选中选项前面的复选框并在选项后面的编辑框输入变量名，就会把相应数据保存到指定的变量中。常用的输出模块为信号与系统模块库（Signals & Systems）中的 Out1 模块和输出方式库(Sink)中的 To Workspace 模块。

3、初始化状态模块

状态模块初始化的方法有两种：使用模块本身的参数设置和从工作空间读入。用于初始化的变量中的元素个数要和状态模块数目一致，而且当从工作空间载入数据时，模块本身的参数设置初始值无效。

7.4 Simulink 模块库

7.4.1 信号源模块库

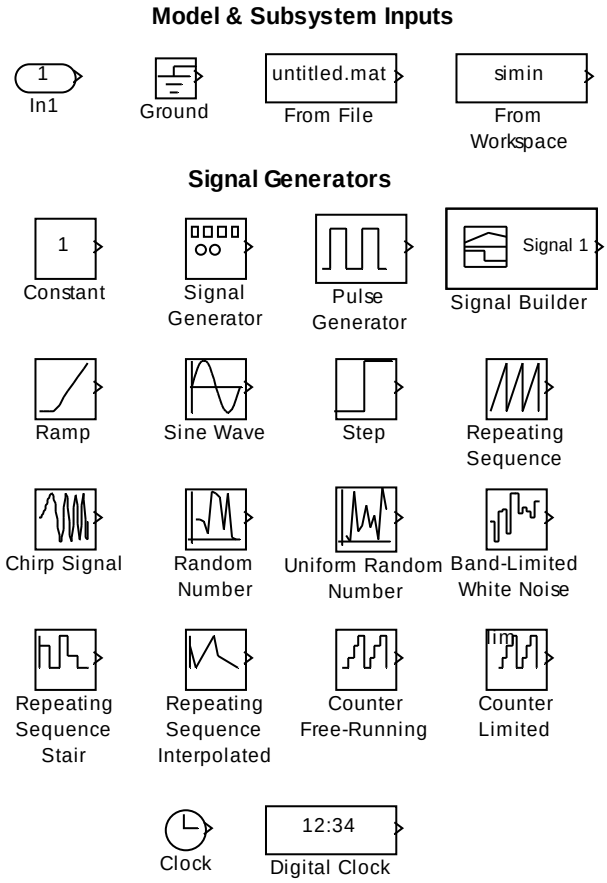

图 7.4.1 Sources 模块库

信号源模块库（Sources）见图 7.4.1，主要包括如下模块：

- 输入端口模块（In1）：用来反映整个系统的输入端子。
- 常数模块（Constant）：可以产生一个常数值，一般用做给定输入。
- 信号发生器(Signal Generator)：可以产生正弦波、方波、锯齿波、随机信号等波形信号，用户可以自由地调整信号的幅值及相位。
- 时钟(Clock)：生成当前仿真时钟，以秒为单位，在记录数据序列或与时间相关的指标中需要此模块。
- 读文件模块(From File)和读工作空间模块(From Workspace)：允许从文件或 MATLAB 工作空间中读取信号作为输入信号。

- 阶跃信号模块(Step)：生成一个按给定的时间开始的阶跃信号，信号的初始值和终值均可自由设定。常用来仿真系统的阶跃响应，也可用来仿真定时的开关动作。
- 其他类型的信号输入模块：带宽限幅白噪声(Band-Limited White Noise)，斜坡输入(Ramp)，脉冲信号(Pulse Generator)，正弦信号(Sine Wave)等。

7.4.2 连续系统模块库

连续系统模块库（Continuous）见图 7.4.2，主要包括如下模块：

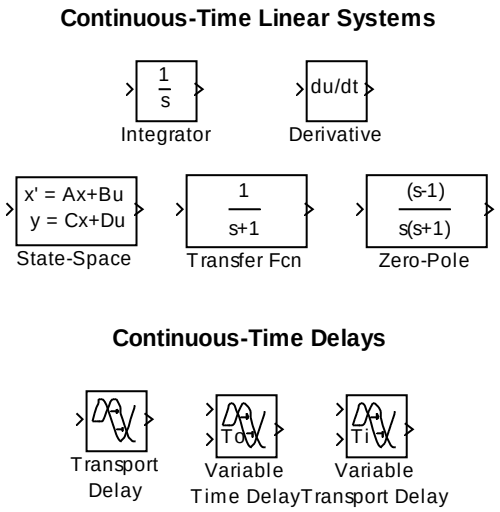

图 7.4.2 Continuous 模块库

- 积分器(Integrator)：对输入（向量或标量）进行积分，用户可以设定初始条件。
- 微分器(Derivative)：将输入端的信号经过一阶数值微分，在输出端输出，在实际应用中应该尽量避免使用该模块。
- 传递函数(Transfer Fcn)：使用分子、分母多项式的形式给出系统的传递函数模型，分母的阶次必须大于或等于分子的阶次。
- 状态空间(State-Space)：使用 A、B、C、D 矩阵形式表示系统的状态空间模型，并可以给出初值。
- 零极点(Zero-Pole)：用指定的零、极点建立连续系统模型。
- 时间延迟(Transport Delay)：将输入信号延迟指定的时间后，再传输给输出信号，用户可自行设置延迟时间。

7.4.3 离散系统模块库

离散系统模块库（Discrete）见图 7.4.3，主要包括如下模块：

- 零阶保持器(Zero-Order Hold)：在一个计算步长内将输出的值保持在同一个值上。
- 一阶保持器(First-Order Hold)：依照一阶插值的方法计算一个步长后的输出值。
- 离散传递函数(Discrete Transfer Fcn)：与连续传递函数结构相同，可设置采样时间。
- 离散状态空间(Discrete State-Space)：与连续状态空间结构相同，可设置采样时间。
- 离散积分器(Discrete-Time Integrator)：实现离散的欧拉积分，可以设置初值和采样时间。
- 离散滤波器(Discrete Filter)：实现 IIR 和 FIR 滤波器。

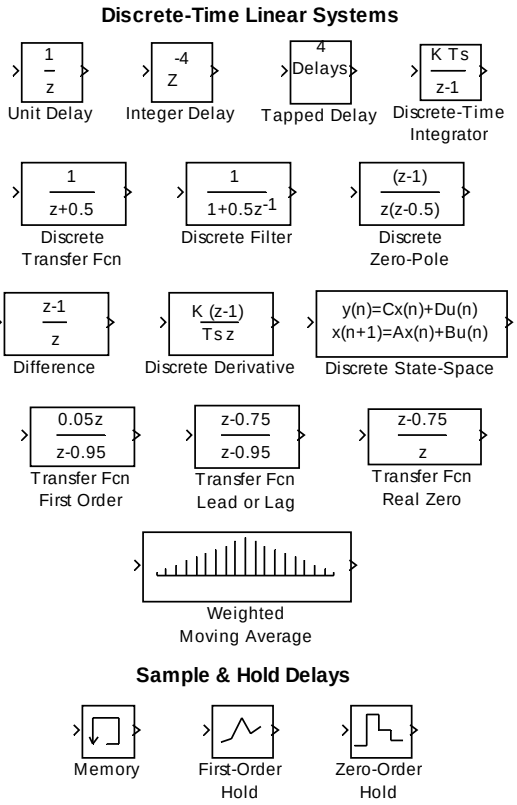

图 7.4.3 Discrete 模块库

7.4.4 数学运算模块库

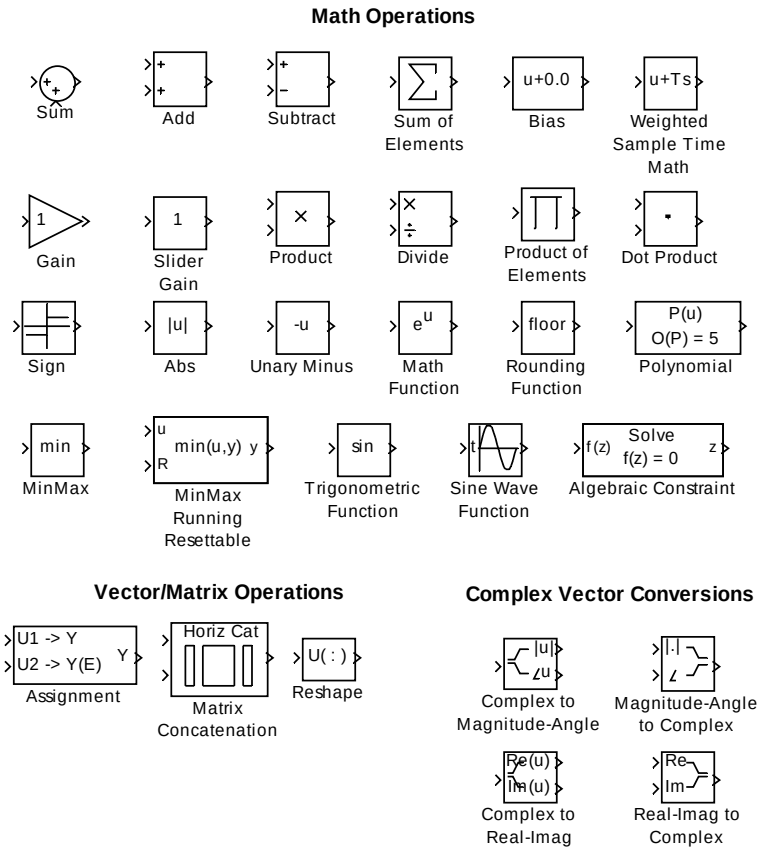

图 7.4.4 Math Operations 模块库

数学运算模块库(Math Operations)见图 7.4.4，主要包括如下模块：

- 增益(Gain)：输出为输入与增益的乘积。
- 加法模块(Sum)：对输入进行求代数和，在组建反馈控制系统方框图时必须采用此模块，反馈的极性（+或-）可自行设置。
- 数字逻辑模块：逻辑运算模块(Logical Operator)，组合逻辑模块(Combinatorial Logic)和位逻辑运算模块(Bitwise Logical Operator)，可以用这些模块很容易地组建数字逻辑电路。
- 数学函数模块：绝对值函数(Abs)，三角函数(Trigonometric)，数学运算函数(Math Function)，复数的实部、虚部提取函数(Complex to Real-Imag)，取整函数(Rounding Function)等。

7.4.5 输出模块库

输出模块库(Sinks) 见图 7.4.5，主要包括如下模块：

- 输出端口模块(Out1)：表示整个系统的输出端，系统直接仿真时该输出将自动在 MATLAB 工作空间中生成变量。
- 示波器(Scope)：显示数据随时间变化的过程和结果。
- x-y 示波器(XY Graph)：将两路输入信号分别作为示波器的两个坐标轴，并将信号的轨迹显示出来。
- 写文件模块(To File)和工作空间写入模块(To Workspace)：将输出信号写到文件或工作空间中。
- 数字显示模块(Display)：将输出信号以数字的形式显示出来。
- 仿真终止模块(Stop Simulation)：强行终止正在进行的仿真过程。

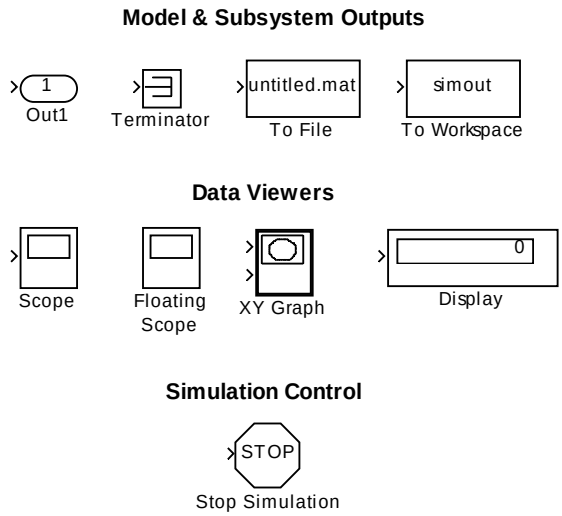

图 7.4.5 Sinks 模块库

7.4.6 非线性系统模块库

非线性系统模块库(Discontinuities)见图 7.4.6，主要包括如下模块：

- 黏性摩擦(Coulomb & Viscous Friction)：在原点不连续，在原点外具有线性增益。
- 滞环非线性模块(Backlash)：该模块具有滞环非线性特性。
- 死区非线性模块(Dead Zone)：该模块具有死区非线性特性。
- 饱和非线性模块(Saturation)：该模块具有饱和非线性特性。

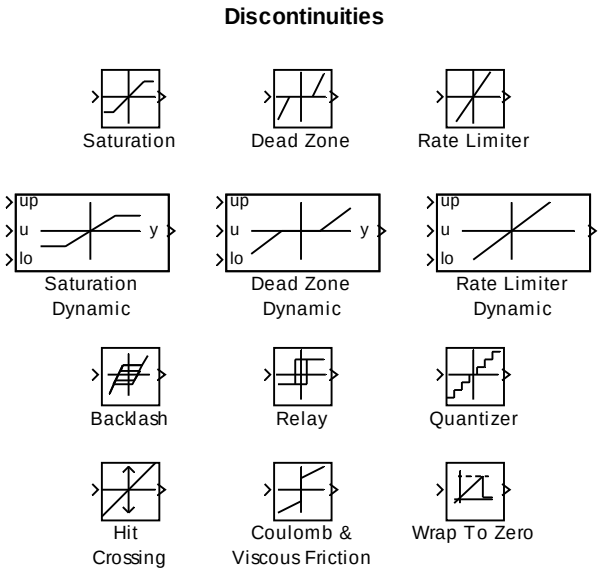

图 7.4.6 Discontinuities 模块库

7.5 Simulink 子系统

7.5.1 Simulink 子系统简介

Simulink 提供的子系统功能可以大大地增强 Simulink 系统框图的可读性，可以不必了解系统中每个模块的功能就能够了解整个系统的框架。子系统可以理解为一种“容器”，我们可以将一组相关的模块封装到子系统模块中，并且等效于原系统模块群的功能，而对其中的模块我们可以暂时不去了解。组合后的子系统可以进行类似模块的设置，在模型的仿真过程中可作为一个模块。建立子系统有以下两种方法：

1、在已有的系统模型中建立子系统

设已有 Simulink 模型图如图 7.5.1 所示，选择需要封装的模块区域（用 Shift 键和鼠标左键配合可以达到同样的目的），框选如图 7.5.2 所示区域，然后右击，弹出浮动菜单，选择【Creat Subsystem】，如图 7.5.3 所示。创建子系统的结果如图 7.5.4 所示。然后双击 Subsystem 模块，弹出子模块如图 7.5.5 所示。

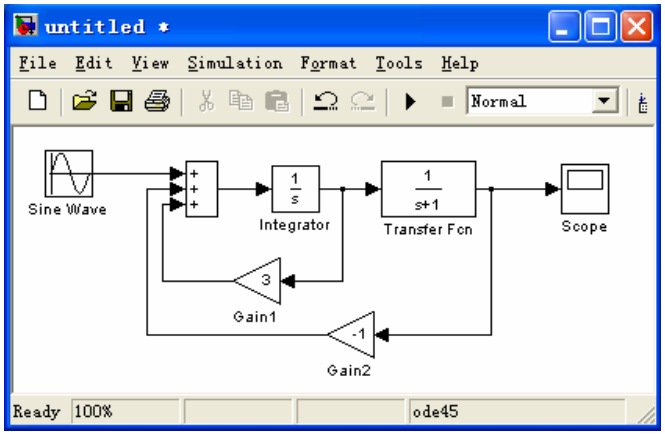

图 7.5.1 简单的 Simulink 模型

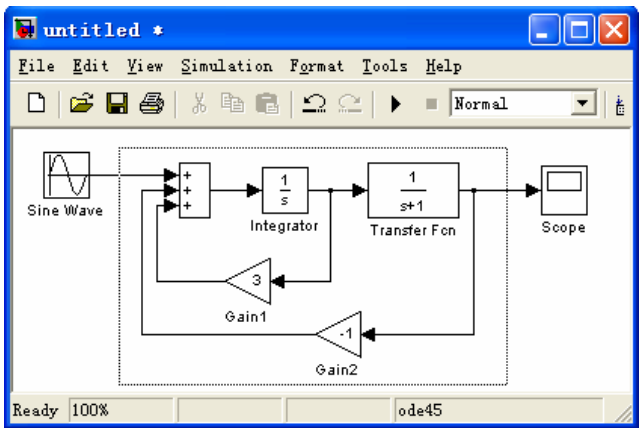

图 7.5.2 框选需要封装的模块

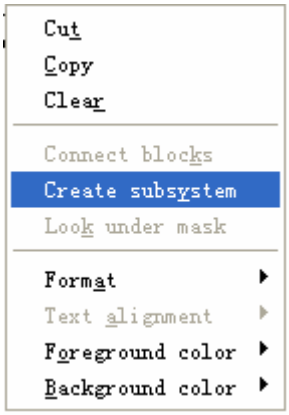

图 7.5.3 快捷菜单

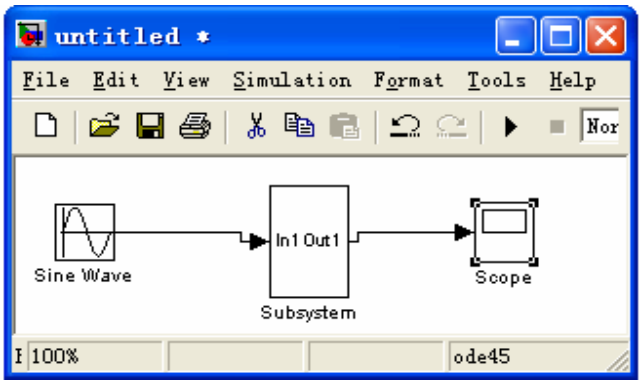

图 7.5.4 创建子系统后的模型

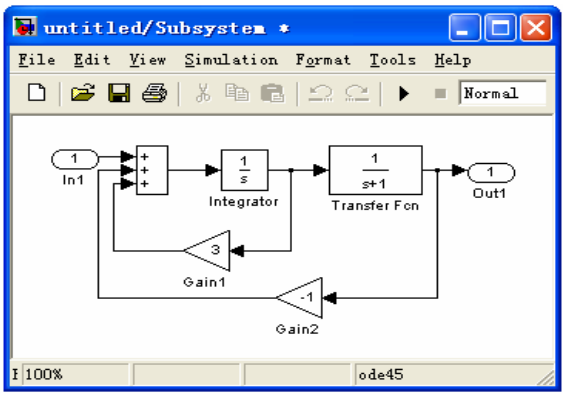

图 7.5.5 子系统模型图

子系统的功能就是把相关模块集中起来，并没有删除。系统的结构仍然没有改变。

2、在已有的系统模型中新建子系统

简单建立模型如图 7.5.6 所示，双击 Subsystem 模块，在弹出的窗口中建立如图 7.5.7 所示模型。

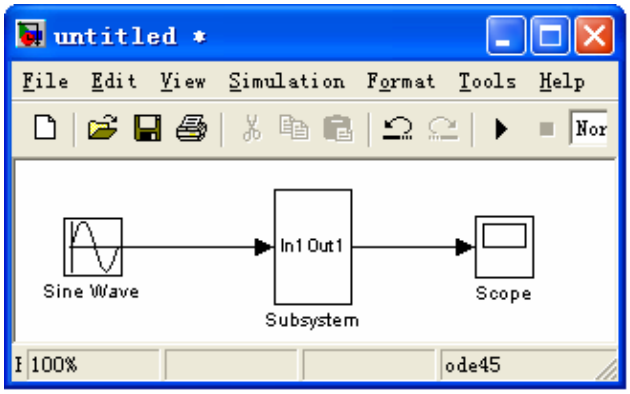

图 7.5.6 含有子系统的模型

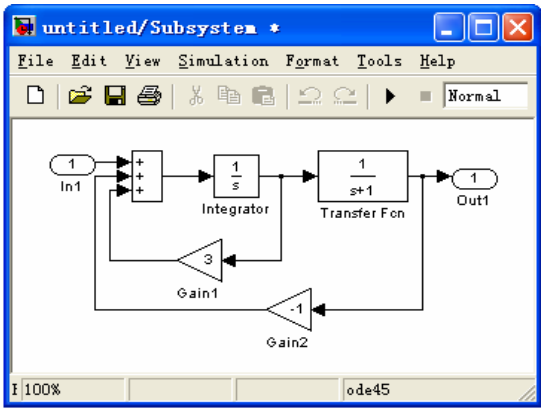

图 7.5.7 子系统模型结构

这两种创建子系统最后实现的是一模一样的功能，只不过操作顺序不同。前者是先将这个结构搭建起来，然后将相关的模块封装起来；后者则是先做一个封装容器，然后在封装容器中添加模块。

对于一个相对简单的模型我们采用第一种，这种操作一般不会出错，能够顺利搭建模型。而对于非常复杂的系统，我们事先将模型分成若干个子系统，然后再采用第二种方法进行建模。

在使用 Simulink 子系统建立模型时，有如下几个常用的操作：

1. 子系统命名：命名方法与模块命名方法类似，是用有代表意义的文字来对子系统进行

命名，有利于增强模块的可读性。

2. 子系统编辑：用鼠标双击子系统模块图标，打开子系统并对其进行编辑。

3. 子系统的输入：使用 Souces 模块库中的 Input 输入模块，即 In1 模块，作为子系统的输入端口。

4. 子系统的输出：使用 Sinks 模块库中的 Output 输出模块，即 Out1 模块，作为子系统的输出端口。

7.5.2 Simulink 高级子系统应用

子系统的最基本目的就是将一组相关的模块包含到一个模块库中，用以简化系统，使得系统的分析更加容易。例如在一个控制系统中，受控系统就可以视为一个子系统，控制器也可以作为一个子系统。从前面的介绍可以发现。这些子系统就如其他一般模块一样，都是具有特定输入输出的模块。对于子系统输入的信号，会产生一个特定的输出信号。但是对于某些特殊的情况，并不是对所有的输入信号都要产生输出信号，只有在某些特定的条件下才会产生输出信号，这就需要输入一个控制信号，控制信号由子系统模块的特定端口输入，这样的子系统称为条件执行子系统。在条件子系统中，输入信号取决于输入信号和控制信号。

根据不同的控制信号，可将条件执行子系统分为如下几类：

1. **触发子系统**：在控制信号满足某种变化要求的瞬间可以触发（激活）子系统，然后保持子系统的输出状态，等待下一个触发信号，它允许用户自己设置在控制信号的上升沿、下降沿或控制信号变化时触发子系统。具体有以下形式：

   (1) 控制信号上升沿触发：子系统在控制信号上升的时候执行。

   (2) 控制信号下降沿触发：子系统在控制信号下降的时候执行。

   (3) 控制信号的双边沿触发：子系统在控制信号符号发生变化时就执行。

2. **使能子系统**：控制信号分成“允许”和“禁止”两种，在允许信号控制下，可以执行子系统中的模块，否则将禁止其功能。为保证整个系统的连贯性，在禁止状态下子系统仍然有输出信号，用户可以选择继续保持禁止前的信号，或复位子系统，强制使其输出零信号。

3. **使能触发子系统**：在使能状态下被触发时将激活该子系统，否则将禁止子系统。

7.5.3 封装子系统

在这里要分清楚封装子系统和建立子系统是两个不同的概念，分别介绍如下：

建立子系统是将一组完成相关功能的模块包含到一个子系统当中，用一个模块来表示，主要是为了简化 Simulink 模型，增强 Simulink 模型的可读性，便于我们仿真和分析。在仿真前，需要打开子系统模型窗口，对其中的每个模块分别进行参数设置。虽然增加了 Simulink 模型的可读性，但并没有简化模型的参数设置。当模型中用到多个这样的子系统，但是每个子系统中模块的参数设置都不相同时，这就显得很不方便。而且容易出错。

为了解决简单建立子系统的不足，我们可以对子系统进行封装。将完成特定功能的相关模块集合到一起，对其中经常要设置的参数设置为变量，然后封装，使得其中变量可以在封装系统的参数设置对话框中统一进行设置，这就大大简化了参数的设置，而且不容易出错，这非常有利于进行复杂的大系统仿真。

封装后的子系统可以作为用户的自定义模块，作为普通模块一样添加到 Simulink 模型中应用，也可以添加到模块库中以供调用。封装后的子系统可以定义自己的图标、参数和帮助文档，完全与 Simulink 其他普通模块一样。双击封装子系统模块，弹出对话框，可进行参数设置，如果有任何问题，可以单击 help 按钮，不过这些帮助是创建者自己进行编写的。

下面对 Simulink 封装子系统的几个特点进行总结：

1. 可以自定义封装子系统的图标；
2. 双击封装后的子系统，弹出参数对话框，其中对话框是自定义的；
3. 封装子系统的帮助文档都是自定义编写的；
4. 封装子系统有自己的工作区域。

以上特点为模型设计带来了很大的方便，具体如下：

1. 将子系统作为一个黑匣子，用户不必了解其中的具体细节而直接使用；
2. 将子系统中模块的参数设置统一到一个参数对话框，大大方便了系统的参数设置；
3. 保护知识产权，防止篡改。

图 7.5.8 是一个 PID 控制器的 Simulink 模型，下面将其封装，以便使图形更简单。

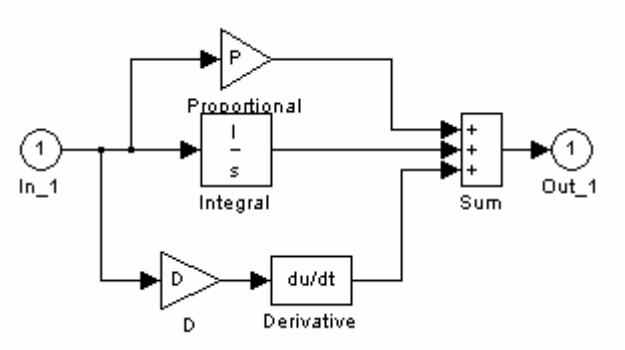

图 7.5.8 PID 控制器模型

首先选中所有模块，点击右键单击 Create Subsystem，生成 Subsystem 模块，然后选中 Subsystem 功能模块，将模块下 Subsystem 名字改为 PID Controller，点击右键选择菜单中的 Mask Subsystem 进入 Mask 编辑窗口，就会出现如 7.5.9 所示的封装编辑器（图 7.5.9 所示是一编辑好的 1con 选项卡）。使用此编辑器可以实现对封装子系统的各种编辑。封装编辑器中共有 4 个选项卡。

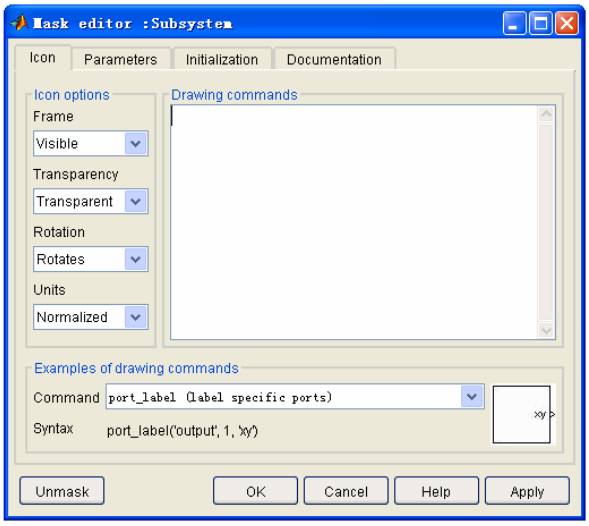

图 7.5.9 封装编辑器

1. 封装编辑器的图标编辑选项卡(1con)

使用图标编辑选项卡用户可以自定义子系统模块的图标。虽然在默认状态下，封装子系统不使用图标，但友好的子系统图标可以使子系统的功能一目了然。用户自定义子系统模块的图标，只需在图标编辑选项卡中的子系统模块绘制命令栏(Drawing commands)中使用绘图命令，并可设置不同的参数控制图标界面的显示。

(1) 图标显示设置(Icon options)

图标边框设置(Frame)：设置图标边框为可见(Visible)或不可见(Invisible)；图标透明性设置（Transparency）：设置图标为透明 Transparency 或不透明 Opaque 显示，图标透明性设置

Transparency 时，图标后面的内容可以显示出来；图标旋转性设置（Rotation）：设置图标固定(Fixed) 或可旋转(Rotates)。图标单位设置（Units）：设置图标绘制命令所使用的坐标系单位，仅对 Plot 和 text 命令有效。其他选项分别为自动缩放(Autoscale)，像素(Pixels)以及归一化表示（Normalized），其中，Autoscale 表示图标自动适合模块大小，与其成比例缩放；Pixels 表示图标绘制用像素作为单位；Normalized 表示模块大小为单位长度，绘制命令中的坐标值不得超过单位值 1 。

(2) 图标绘制命令栏（Drawing commands）

封装后的子系统模块的图标均是在图标绘制命令栏中完成绘制的。使用不同的绘制命令可以生成不同的图标。生成的图标可以是描述性文本、子系统数学模型图标、图像或图形等，如果在此栏中键入多个绘制命令，则图标的显示按照绘制命令的顺序显示。

- 描述性文本图标

使用下列命令可以在模块图标上显示文本：

```
disp('text')            %图标上显示 text 成文本字样。
disp(variablename)   % variablename 为工作空间中的字符串变量名。
Text(x, y, 'text')       %在图标上特定位置显示 text 成文本字样。
text(x，y，stringvariablename) %stringvariablename 为已存在的字符串变量名。
fprintf('text')
fprintf ('format' variablename)     % format 表示文本的格式。
port_label(port_type，port_number，lable) %此命令可以显示模块的端口名称，其中
                                          port_type 为端口类型，取值为'input'或'output'，
                                          port_number 为端口数目，1abel 为端口文本。
```

如果需要显示多行文本，可以使用＼n 换行。这时封装后的子系统图标为描述性文本。

- 子系统数学模型图标

使用 dpoly 命令可以将封装的子系统模块的图标设置为系统的传递函数，使用 droots 命令可以设置为零极点传递函数，其命令格式为

```
dpoly(num，den)
dpoly(num，den，'character')
droots(z，p，k)
```

其中，num，den 分别是传递函数的分子和分母多项式；'character'(取 s 或 z)为系统的频率变量；z，p，k 分别是传递函数的零点、极点和系统增益。需要注意的是，参数 num，den，z，p，k 必须是工作空间中已经存在的变量，否则绘制命令的执行将出现错误。

- 图像或图形图标

使用 plot 或 image 命令可以将子系统模块的图标设置为图形或图像。其命令为

```
plot(x，y)
image(imread('photoname'))
```

(3) Examples of drawing commands

该选项组说明了不同的 Simulink 支持的图标绘图命令。为了能够了解其中命令的语法，可以从命令菜单中选择相应的命令，Simulink 就会显示所选择命令的实例，并会在右下角产生一个图标。

本例中设置参数如图 7.5.10 所示，在绘制命令窗口中键入 disp（'PID'），其他参数下拉菜单如图中所示。

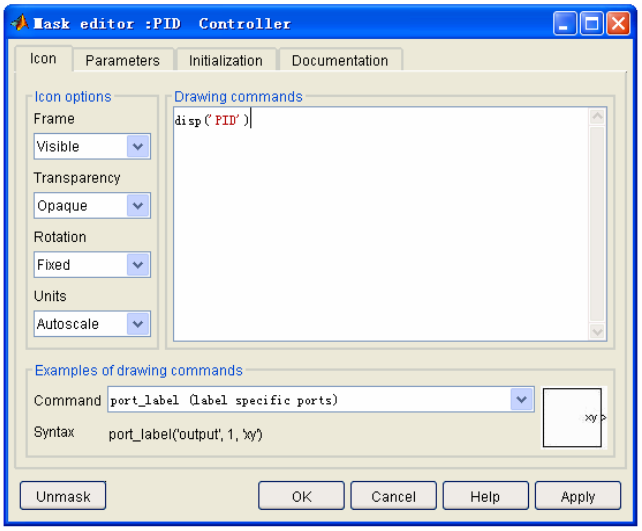

图 7.5.10 封装编辑器（Icon）选项卡

2. 封装编辑器的参数设置选项卡（Parameters）

子系统封装的目的之一是提供一个友好的参数设置界面，通常用户不需要了解系统内部的细节，只需要提供正确的模块参数即可完成对系统的设计与仿真。只有使用了子系统封装编辑器中提供的参数选项卡（Mask Editor 下的 Parameters 选项卡）进行子系统参数设置，才可以说是真正完成了子系统的封装，从而使用户设计出与 Sinulink 模块库中同样直观的参数设置界面。

不同于通常的子系统，封装的子系统具有独立的工作空间，这是由于在没有对子系统进行封装之前，子系统中的模块可以直接使用 MATLAB 工作空间的变量，通常的子系统可以看作是图形化的 MATLAB 脚本，即子系统只是将一些由模块实现的命令以图形化的方式组合而成的。而封装的子系统的内部参数对系统模型中的其他系统不可见，而且只能使用参数设置选项卡输入。

参数设置选项卡中包括如下几种设置内容：

1) 参数设置控制

参数设置控制包括添加（Add）、删除（Delete）、上移（Move up）和下移（Move Down），分别表示在即将生成的参数设置界面中添加、删除、上移与下移模块需要的输入参数。

2) 参数描述

参数描述是对模块输入的参数作简要的说明，在子系统参数设置界面中用来区别不同的参数 ，因而其取值最好能够说明参数的意义或作用。

3) 变量

用来指定键入的参数值将要传递给的封装子系统工作空间的相应变量，此处使用的变量必须与子系统中所使用的变量相同。

4) 参数设置描述

参数设置描述包括参数控制类型（Type）、是否为求值字符串（Evaluate）、是否可调整（Tunable）复选框，其中控制类型包括 Edit（需要用户在参数设置界面中键入参数值，适合多数情况）、Checkbox（复选框，表示逻辑值）和 Popup（在参数设置界面中弹出参数选项以便选择参数，弹出的参数选项值在 Popup 栏中输入）。

在本例中，对 PID 控制器进行封装，这里针对该例说明如何对封装后的子系统的参数设置选项卡进行设置。在封装子系统模块的参数设置界面中，应该提供控制律所需的 P、I、D 参数，如图 7.5.11 所示。

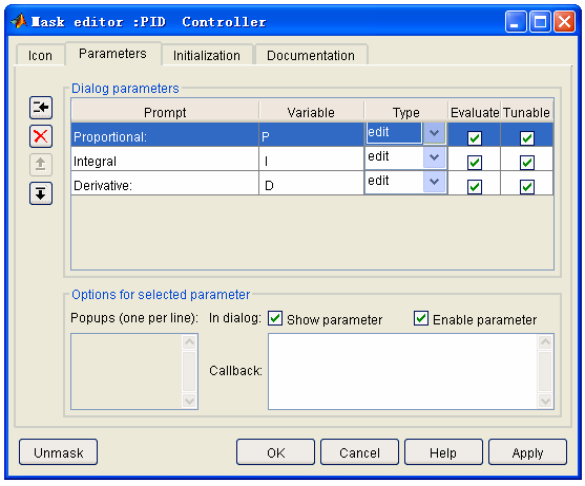

图 7.5.11 封装子系统模块系统设置选项卡

如图 7.5.11 所示，对封装子系统进行参数设置后，双击封装的子系统即可打开子系统的参数设置界面如图 7.5.12 所示，用户只需输入正确的参数即可进行系统的仿真设计分析。

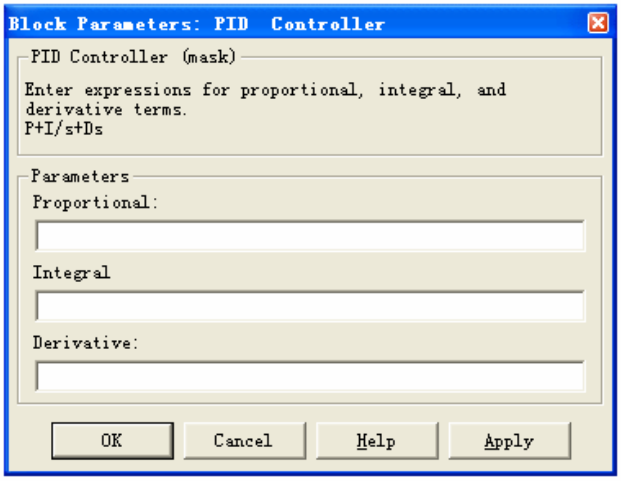

图 7.5.12 封装后子系统得参数设置界面

3. 封装编辑器的初始化设置选项卡（Initialization）

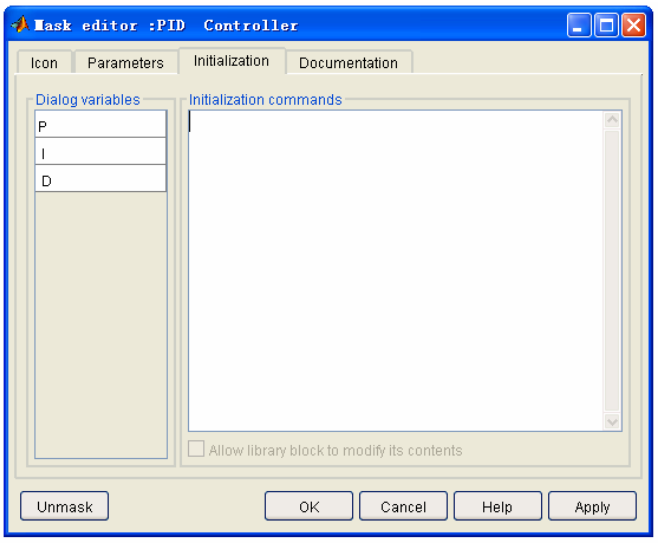

图 7.5.13 封装编辑器的初始化设置选项卡

初始化设置选项卡中的对话变量表是用户设置了参数设置选项卡后自动生成的。初始化命令一般为 MATLAB 命令，在初始化命令栏中可以定义封装后子系统工作空间中的各种变量，这些变量可以被封装子系统模块图标绘制命令、其他初始化命令或子系统中的模块使用。初始

化命令要用分号来结尾，避免在 MATLAB 命令窗口中出现回调结果。如果配合在 Initialization commands 内编辑程序，则可以发挥功能模块的功能来执行特定的操作。Initialization 选项卡如图 7.5.13 所示。Dialog variables 选项组中的列表显示了与封装子系统参数相关的变量名，用户可以从这个列表中复制参数名到 Initialization commands 框中，也可以使用这个列表来更改参数变量，用鼠标双击相应的变量就可以更改了，然后按回车确定。 Allow library block to modify its contents 复选框仅当封装子系统存在于模块库中才可用，选中这个复选框允许模块的初始化代码修改封装子系统的内容。

4. 封装编辑器的文档编辑选项卡（Documentation）

Simulink 模型库中的模块均提供了模块的简单描述和详细的帮助文档，方便用户的使用和理解，对于用户封装的子系统模块，封装编辑器的文档编辑选项卡可以使用户建立被封装子系统的所有帮助文档。Documentation 选项卡如图 7.5.14 所示。

- Mask type：为封装子系统模块参数对话框设置说明的标题，仅供用户来对文档进行分类，它出现在模块参数对话框中和所有的模块封装编辑窗口中，用户可以选择喜欢的名字，当 Simulink 创建模块对话框时，它会自动在后面添加 mask，以区别系统内建的模块。
- Mask description：对封装子系统模块的工作进行说明，可以在说明中使用 Enter 和 Space 键，此参数中要尽量对模型进行描述，以便用于其他模型当中。
- Mask help：该模块的帮助文档，可以单击封装子系统中的 Help 按钮进行查看，这可以讲述如何对模块的参数进行设置，以便用于其他模型当中。

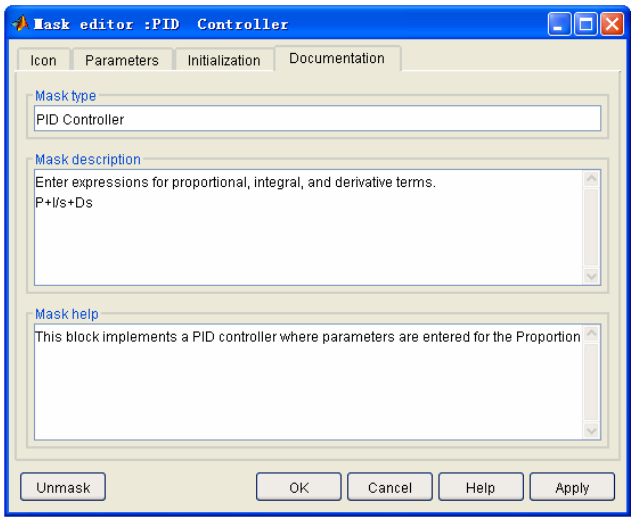

图 7.5.14 封装编辑器的文档编辑选项卡

按照上述步骤对 PID 模型进行封装后得到如下图 7.5.15 所示模型

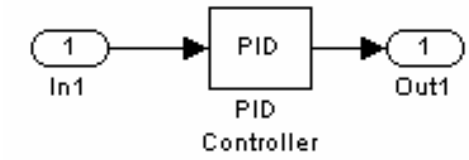

图 7.5.15 封装后的模型

7.6 S 函数的建立

S 函数就是 S-Functions，是 System-Functions 的缩写。当 MATLAB 所提供的模型不能完全满足要求时，就可以通过 S 函数提供给用户编写程序来满足要求模型的接口。S 函数使用有固定格式，只能用于基于 Simulink 的仿真，并不能将其转换成独立于 MATLAB 的程序。S 函数功能非常全面，适用于连续、离散以及混合系统。

7.6.1 S 函数的调用格式

MATLAB 提供了编写 S 函数的模板文件，该模板文件位于 MATLAB 根目录 toolbox\Simulink\blocks 下，文件名为 sfunmpl.m，该文件给出了 S 函数完整的框架结构，包含一个主函数和若干子函数，每一个子函数都对应于一个 flag 值，用户还可以根据编程需要加以修正。

用 MATLAB 语言编写 S 函数的调用格式为：

Function [sys,x0,str,ts]=sfun(t,x,u,flag,p1,p2,…)

其中 sfun 是 S 函数的函数名，t,x,u 分别为时间、状态和输入信号，flag 为标志位，p1,p2 为参数。

flag 标志位为不同值时调用子函数和功能如表 7.6.1 所示。

表 7.6.1 S 函数的 flag 值表

Flag 值	仿真过程	调用的子函数	功能
0	初始化	mdlInitializeSize()	对连续/离散状态变量个数、输入和输出的路数、采样周数个数和值以及状态变量初值 x0 等进行初始设置。
1	连续状态变量更新	mdlDerivatives ()	计算连续状态变量的微分方程并由 sys 变量返回
2	离散状态变量更新	mdlUpdate()	更新离散状态变量并由 sys 变量返回
3	计算输出	mdlOutputs()	计算输出信号由 sys 变量返回
4	计算下一仿真时刻	mdlGetTimeNextVarHit ()	计算下一步的仿真时刻由 sys 变量返回
9	终止	mdlTerminate()	终止仿真过程，不返回任何变量

7.6.2 S 函数的模板格式

S 函数的模板格式为：

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;     %调用初始化子函数
  case 1,
    sys=mdlDerivatives(t,x,u);              %调用计算导数子函数
```

```
  case 2,
    sys=mdlUpdate(t,x,u);              %调用离散状态变量更新子函数
  case 3,
    sys=mdlOutputs(t,x,u);             %计算输出子函数
  case 4,
    sys=mdlGetTimeNextVarHit(t,x,u); %计算下一仿真时刻子函数
  case 9,
    sys=mdlTerminate(t,x,u);           % 终止仿真子函数
  otherwise
    error(['Unhandled flag = ',num2str(flag)]); % flags 为其他值提示错误信息
end
```

1、初始化子函数

```
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates= 0;      %设置连续状态变量个数，默认值为 0
sizes.NumDiscStates= 0;      %设置离散状态变量个数，默认值为 0
sizes.NumOutputs= 0;         %设置输出变量的个数，默认值为 0
sizes.NumInputs= 0;          %设置输入变量的个数，默认值为 0
sizes.DirFeedthrough = 1;    %输入信号是否直接在输出端出现，取值为 0 或 1
sizes.NumSampleTimes = 1; %设置采样周期的个数，默认值为 1
sys = simsizes(sizes);
x0   = [];                   %设定初始值，默认值为[]
str = [];
ts   = [0 0];                % [0 0]用于连续系统，[-1 0]表示继承其前的采样时间设置
```

2、计算导数子函数

```
function sys=mdlDerivatives(t,x,u)
sys = [];                    %计算结果由 sys 变量返回
```

3、离散状态变量更新子函数

```
function sys=mdlUpdate(t,x,u)
sys = [];                    %更新后的离散状态变量由 sys 变量返回
```

4、计算输出子函数

```
function sys=mdlOutputs(t,x,u)
sys = [];                    %计算的输出向量由 sys 变量返回
```

5、计算下一仿真时刻子函数

```
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;              %设置本函数的调用时间
sys = t + sampleTime;        %计算下一采样时刻由 sys 变量返回
```

6、终止仿真子函数

```
function sys=mdlTerminate(t,x,u)
sys = [];
```

【说明】在实际运用时，并不是都要调用所有的子函数，用户可根据实际需要去掉某些值；输入参量的设置可在调用此 S 函数时给定。

例 7.6.1 应用 S 函数实现增益，使得 y=4*u。

解：1、用 M 文件建立文件名为 sfunl.m 的 S 函数

```
function [sys,x0,str,ts]=sfunl(t,x,u,flag)
switch flag,
```

```
    case 0                                  %初始化
[sys,x0,str,ts]=mdlInitializeSizes;
    case 3                                  %计算输出
sys=mdlOutputs(t,x,u);
    case {1,2,4,9}                          %非执行的 flags 值
sys=[ ] ;
     otherwise                              %错误处理
     error (['Unhandled flag=',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes()
sizes=simsizes;
sizes.NumContStates=0;
sizes.NumDiscStates=0;
sizes.NumOutputs=-1;
sizes.NumInputs=-1;
sizes.DirFeedthrough=1;
sizes.NumSampleTimes=1
sys=simsizes(sizes);
str=[ ];
x0=[ ];
ts=[-1 0];
function sys=mdlOutputs(t,x,u)
sys=4*u;
```

2、在 Simulink 环境下建立名为 smodel.mdl 的模型

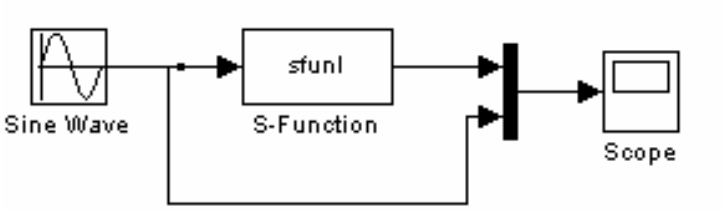

图 7.6.1 smodel.mdl 模型仿真图

其中，S-Functions 模块参数设置如图 7.6.2 所示，Sine Wave 模块中将 Frequency 设置为 2

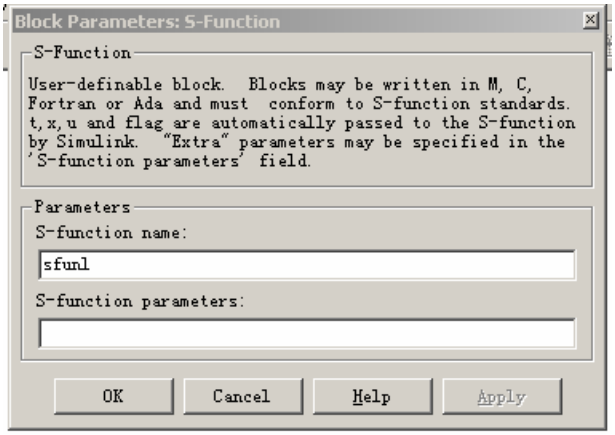

图 7.6.2 S 函数参数设置对话框

3、运行 smodel 模型，得到如图 7.6.3 所示仿真曲线

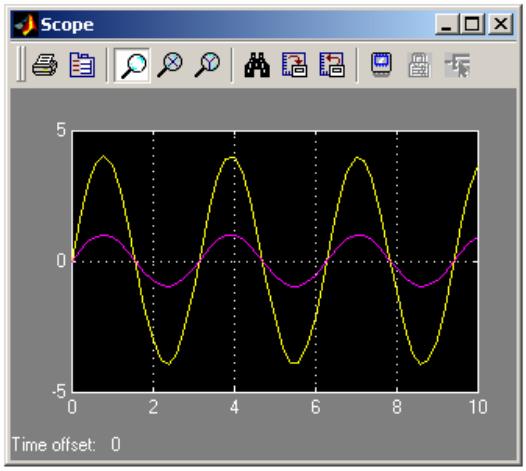

图 7.6.3 S 函数运行曲线图

小结

作为 MATLAB 的重要组成部分，Simulink 具有相对独立的功能和使用方法，确切地说它是对基于信号流图的动态系统进行仿真、建模和分析的软件包，它不但支持连续、线性系统仿真，而且也支持离散、非线性系统仿真。

Simulink 提供了对系统信号流图进行组态的仿真平台，通过 Simulink 模块库建立系统的仿真模型，可直观、方便地对系统进行动态仿真。

子系统和 S 函数的使用扩展了 Simulink 系统仿真的功能，提高了其使用性能，使 Simulink 仿真的优势得以充分发挥。