第 5 章 Simulink建模与仿真技巧

CHAPTER 5

Developing Modeling and Simulation Skills in Simulink

第 4 章介绍了简单 Simulink 模型的绘制方法和系统仿真的基本流程。本章将继续深入探讨基于 Simulink 的建模与仿真技术，涵盖从模型构建、仿真优化到结果可视化与模块封装的全流程方法，旨在帮助读者掌握更为复杂和实际的仿真建模技能。本章内容组织如下：

5.1 节介绍常用建模应用技巧，包括向量化模型的构建方法、线性系统的仿真建模、代数环现象的检测与消除、过零点检测的重要性以及仿真快速重启技术。这些技巧是提升建模效率与仿真精度的关键。5.2 节聚焦于非线性环节与查表环节的构建，讲解如何利用查表模块和开关模块搭建单值与双值非线性模块，并介绍利用 MATLAB 代码模块描述一般静态非线性函数的方法，以增强模型的灵活性与表达能力。5.3 节系统性地介绍各类微分方程的 Simulink 建模与求解方法，涵盖一般微分方程、微分代数方程、延迟微分方程、切换系统以及分数阶微分方程。本节侧重于通过实例演示仿真模型的搭建逻辑，并提供通用的仿真框架。5.4 节探讨输出模块的使用与结果可视化。内容包括信号查看器、表盘显示、数字信号处理，并介绍 Simulink 模型的布线技巧。5.5 节介绍如何利用虚拟现实技术实现仿真结果的三维动画展示，简要介绍虚拟现实的场景模型创建方法，并通过实例展示虚拟现实场景的驱动与显示。5.6 节详细讲解子系统的概念与模块封装技术。内容包括子系统的创建方法、子系统模块的封装技术、自定义模型库的构建方法，并介绍子系统的控制与应用。通过一个复杂系统的案例，展示子系统在大型仿真建模中的实际价值。

通过本章的学习，读者将能够掌握 Simulink 中级建模与仿真技术，具备解决复杂工程问题和实现高效系统仿真的能力。

5.1 常用建模应用技巧

前面介绍过 MATLAB 的向量化编程技术，事实上，Simulink 建模也可以使用向量化技术，更高效地建立仿真模型。另外，本节还将探讨模块排序与代数环消除、过零点检测技术与仿真模型的快速重启技术等。

5.1.1 向量化模块举例

在当前版本的 Simulink 中，很多模块不但支持标量型信号的输入，还支持向量型的信号输入，即将多路的信号通过 Mux 模块合成一路向量型信号，从而直接输入给这类模块。下面将通过几个例子来演示向量化的模块及其应用。

例5-1　试用向量化方法重新建立例4-2中曾给出了van der Pol方程的仿真模型。

解　例4-2给出的Simulink底层模型结构很烦琐。事实上，若将其状态方程写成下面的向量形式：

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix} x_2(t) \\ \mu(x_1^2(t)-x_2(t))x_1(t)-x_2(t) \end{bmatrix}$$

则可以用单个积分器模块来完成向量化建模，得出的Simulink模型如图5-1所示。注意，在向量化建模中，可以将向量的各个组成信号通过Demux模块进行分解，然后可以单独对各路信号进行运算，再将结果用Mux模块汇合成单路的向量型信号，传输给向量化的模块进行处理。

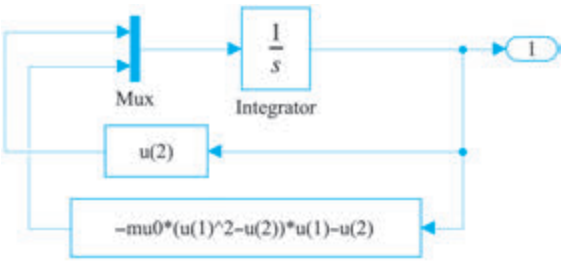


图5-1　信号向量化的Simulink模型（文件名：c5mvdp1a.slx）

在这个系统下只需将积分器模块的初值也设置为列向量的形式，即可对原系统进行仿真，得出的结果将与例4-2完全一致。

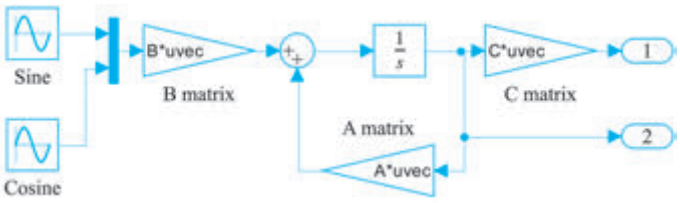
模型中使用了两个Fcn模块，其原因是，该模块只能支持标量输出，不支持向量化输出。如果处理高阶微分方程，则需要更多的Fcn模块，使用起来很麻烦，所以需要支持向量甚至矩阵输出的模块。后面将介绍更简洁的建模方法。

例5-2　假设双输入双输出系统的状态方程表示为：

$$\dot{\boldsymbol{x}}=\begin{bmatrix} 2.25 & -5 & -1.25 & -0.5 \\ 2.25 & -4.25 & -1.25 & -0.25 \\ 0.25 & -0.5 & -1.25 & -1 \\ 1.25 & -1.75 & -0.25 & -0.75 \end{bmatrix}\boldsymbol{x}+\begin{bmatrix} 4 & 6 \\ 2 & 4 \\ 2 & 2 \\ 0 & 2 \end{bmatrix}\boldsymbol{u},\ \boldsymbol{y}=\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2 \end{bmatrix}\boldsymbol{x}$$

且输入信号分别为$\sin t$和$\cos t$。试建立Simulink仿真模型。

解　前面介绍过，Simulink提供了线性系统的状态方程模块，但该模块并不能直接得出系统的内部状态，所以可以用Simulink模块搭建起带有状态变量输出的新状态方程模型，如图5-2(a)所示。注意，因为这里需要的是矩阵增益，而不是普通增益，所以应该双击增益模块，打开的界面如图5-2(b)所示，可以从中选择“矩阵K*u(u向量)”选项，而不能选择一般的标量增益模块。另外，在模型设计时并不存在任何局限性，所以这样建立起来的模型应该适合于任意的连续多变量线性系统。



（a）状态方程（文件名：c5mss1.slx）

（b）增益模块参数设置列表框

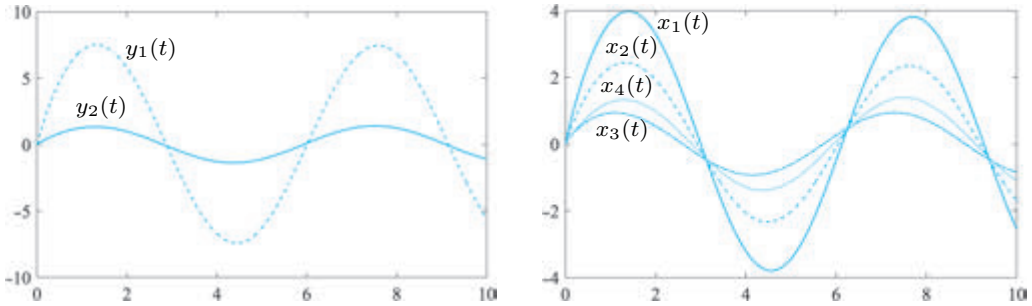
图5-2　带有状态变量输出的状态方程模型

假设系统的初始状态向量$\boldsymbol{x}_0$为零向量，将其写入积分器模块，并给各个矩阵赋值，将这些语句写入模型的PreLoadFcn属性，以便模型启动时能自动导入参数。

```
>> A=[2.25,-5,-1.25,-0.5; 2.25,-4.25,-1.25,-0.25;
      0.25,-0.5,-1.25,-1; 1.25,-1.75,-0.25,-0.75];
   B=[4,6; 2,4; 2,2; 0,2]; C=[0,0,0,1; 0,2,0,2];
   D=[0,0; 0,0]; x0=zeros(4,1);
```

对整个系统进行仿真，则可以得出tout和yout两个变量，其中yout的前两列为系统的输出信号，后4列为系统的状态变量，用下面语句则可以得出如图5-3所示的曲线。

```
>> plot(tout,yout(:,1:2));          % 系统的输出曲线
   figure; plot(tout,yout(:,3:6)) % 系统的状态曲线
```



（a）输出信号　　（b）状态变量信号

图5-3　双输入双输出系统的仿真结果

事实上，如果想利用Simulink自身的状态方程模块，还同时想获得系统的状态变量，则可以增广$\boldsymbol{C}$和$\boldsymbol{D}$矩阵，使之等于：

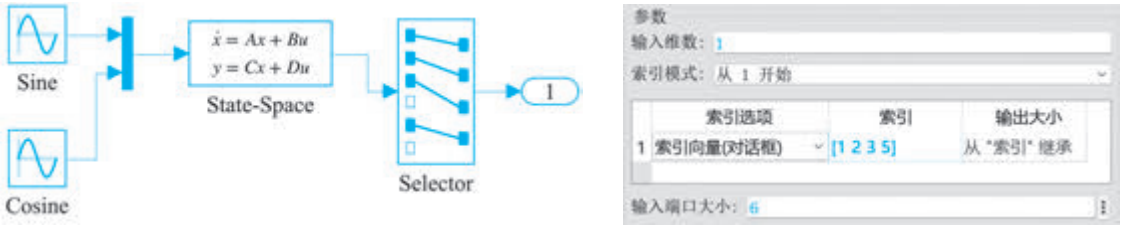
```
>> C=[C; eye(4)]; D=[D; zeros(4,2)];
```

这样修改后的状态方程模型将有6路输出，前两路为实际输出信号，后4路为系统的4个状态变量。这样可以将连续相同模块组中的状态方程模块嵌入仿真模型，在其后连接Demux模块并将其参数设置为[2,4]，该系统得出的结果将和上面的完全一致。

信号与系统模块组中的Selector（选路器）模块允许用户从向量信号中有选择地提取一些信号，构成新的向量信号。下面将通过例子演示该模块的使用方法。

例5-3　考虑例5-2中带有状态输出的状态方程模型。如果想从输出端口中提取系统的两路输出与第1、3状态变量，试利用Selector模块完成信号的选择。

解　可以建立起如图5-4(a)所示的仿真框图。双击选择器模块，将打开如图5-4(b)所示的对话框，可以在“输入端口大小”编辑框中输入6（总输入路数，该数值必须和实际输入信号一致，否则将出现错误），在“索引”编辑框填写输入到输出的对应关系。

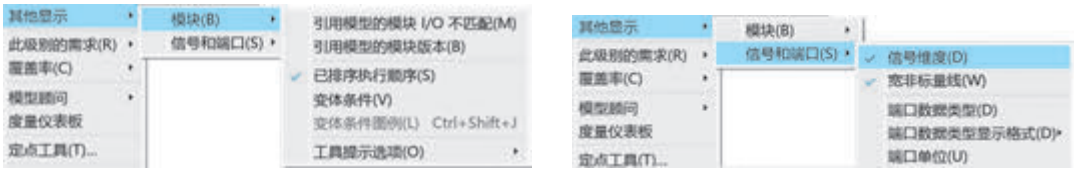


（a）Simulink模型（c5mss2.slx）　　（b）选择器参数设置

图5-4　带有选择器模块的系统

5.1.2 Simulink模型的信号标识与排序

为了增加Simulink模型的可读性，便于了解系统的内部特征，Simulink环境中提供了若干对模型表示的修饰选项。例如，模型快捷菜单的“其他显示”（R2026a版开始改为“叠加信息”）菜单项提供了如图5-5所示的子菜单项。



（a）模块子菜单　　（b）信号与端口子菜单

图5-5 “其他显示”快捷菜单

通过这样的子菜单，可以考虑用更好的方式显示Simulink模型。

- **模块运行排序**。在Simulink模型进行仿真前，会对每个模块自动进行排序。一般情况下，将输入源和积分器（因为初值已知）自动地排在最前面，然后开始排序其他派生信号。选择“其他显示 ▸ 模块 ▸ 已排序执行顺序”子菜单项，再执行一次仿真，则将在每个模块附近显示该模块在仿真框图中的排序序号，如图5-6所示。

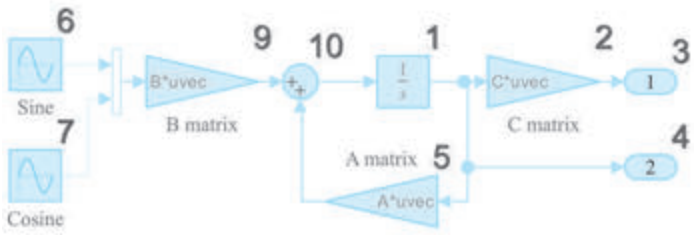


图5-6 Simulink模型的模块排序

可见，由于积分器初值已知，所以将该模块排为1号，由其驱动，依次运行2号、3号、4号、5号模块。5号模块的信号进入加法器，但不能直接运行加法器模块，因为加法器的另一路输入信号暂时未知。现在考虑已知的正弦、余弦信号模块，可以自然地将其排位6号、7号，由它们可以计算出Mux信号，排为8号模块（图中未标出），驱动B matrix模块，排为9号。这样就生成了加法器的另一路输入信号。现在，加法器两路输入信号都已知，所以可以将加法器排为10号模块，因此完成整个仿真回路的计算顺序，下一步仿真依照这样的顺序再推演一步。用这样的方式可以完成整个仿真进程。

- **信号和端口显示**。选择“其他显示 ▸ 信号与端口 ▸ 信号与维度”菜单项，则可以将各个信号的维度模型中显示出来；选择“其他显示 ▸ 信号与端口 ▸ 宽非标量线”菜单项，则可以将向量化的信号用粗线表示出来。考虑图5-2（a）中给出的Simulink模型，选择“加粗”和“信号路数”命令后将得出如图5-7所示的图形效果。还可以选择“其他显示 ▸ 信号与端口 ▸ 端口数据类型”菜单项，将信号的数据类型显示出来，如用double显示最常用的双精度数据，而double(2)表示两路向量化的双精度信号。

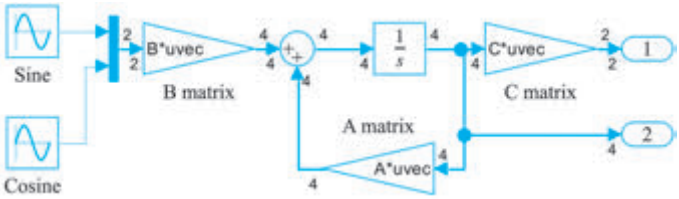


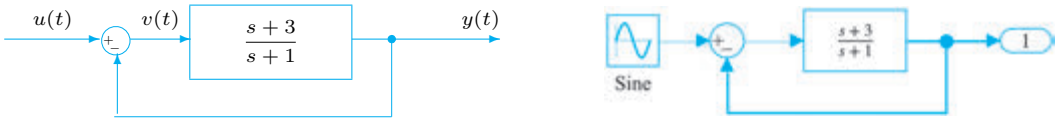
图5-7 向量信号线的修饰（文件名：c5mss1a.slx）

5.1.3 Simulink的代数环及消除方法

代数环（algebraic loop）是指在一个仿真模型中，某个或某些模块的输出信号取决于其输入信号，而输入信号同时又取决于其输出信号的现象。这需要在每步仿真过程中，求解一次代数方程。由于代数方程的求解是比较耗时的，带有代数环的仿真模型势必增加仿真过程的计算量。本节通过例子演示代数环的现象，然后给出一种有效的代数环避免方法。

例5-4 考虑图5-8(a)中给出的一个简单的线性反馈系统模型，试构造Simulink仿真模型。

解 可以搭建如图5-8(b)所示的Simulink仿真框图。如果将“模型设置”对话框“诊断 ▸ 代数环”列表框设置为“错误”，则仿真时如果遇到代数环，则仿真自动停止，并将涉及代数环的几个模块用红色显示出来。对这样的系统进行仿真，加法器、传递函数模块构造的函数变成红色，说明这个Simulink框图存在代数环。可以看出传递函数模块的输入为 $v(t)=u(t)-y(t)$，而 $y(t)$ 又是它自己的输出，这就形成了一个怪圈，这个怪圈就是所谓的代数环现象。



（a）反馈系统框图　　（b）Simulink仿真模型（文件名：c5malgb.slx）

图5-8 简单反馈系统模型

在快捷菜单中选择“其他显示 ▸ 模块 ▸ 已排序执行顺序”，则可以对模型模块进行排序，得到如图5-9所示的排序结果。可见，“2.1”和“2.2”模块无法正常排出执行次序，形成一个代数环。

图5-9 仿真模型的模块排序

例5-4给出的代数环是由两个环节构成的，比较容易找到。在一些其他应用中，代数环可能由很多模块组成的回路构成，这就需要花一些时间利用给出的错误或警告信息去发现哪些模块是代数环的组成部分。

为保障仿真过程的正常执行，通常Simulink将代数环现象设置为警告处理信息，而不是错误信息。这样，在仿真问题求解过程中，每一步仿真都需要花一定的时间求解代数环对应的代数方程，这会减慢整个仿真过程。对一些系统来说，忽略代数环是没有问题的，而另一些系统则不能忽略代数环，所以应该考虑积极的方法避免代数环。

对这个具体例子而言，可以直接计算出闭环模型$G_1(s)=G(s)/(1+G(s))$，得出等效模型$(s+3)/(2s+4)$，再由$G_1(s)$模块建立Simulink仿真模型，完全消除代数环。然而，绝大部分系统的代数环是不能完全消除的。所以建议使用Simulink的机制运行仿真进程，如果只得出警告信息，则可以接受得到的仿真结果；如果Simulink因此给出错误信息，不能完成仿真进程，则有必要引入近似的方式消除代数环。

现在以图5-8（a）为例分析代数环产生的原因，再探讨近似消除代数环的方法。在该图描述的计算过程中，因为计算$y(t)$需要已知$u(t)$，而计算$u(t)$又同时需要$y(t)$，所以产生了代数环。如果消除这个“同时”的关系，在反过来计算$u(t)$时，不需要$y(t)$，而需要一个$y(t)$的近似信号，则可以消除代数环。如何生成$y(t)$的近似信号呢？给$y(t)$加一个微小的延迟是一个显然的选择，不过仿真实例证明这种方法可能引入更大的误差[1]。另一种方法是在$y(t)$后面接一个低通滤波器$1/(Ts+1)$，T为微小的正数，得出$y_1(t)$。在计算$u(t)$时使用$y_1(t)$而不是$y(t)$，则可以消除代数环。只要T的值远小于原系统的时间常数，代数环的作用可以完全忽略。

例5-5 试引入低通滤波器消除例5-4系统的代数环，并评价处理的效果。

解 由给出的等效传递函数可以反推回系统的微分方程

$$\dot{y}(t)+4y(t)=\dot{u}(t)+3u(t),\ y(0)=0.5$$

其中，$u(t)$为阶跃函数或Heaviside函数。这样，可以使用下面的方法直接求取微分方程的解析解。

```
>> syms t y(t); u=heaviside(t);
   y0=simplify(dsolve(2*diff(y)+4*y==diff(u)+3*u, y(0)==0.5))
```

可以得出输出信号的解析解为

$$y_0(t)=\frac{1}{8}\mathrm{e}^{-2t}+\frac{3}{8}\mathrm{sign}\,(t)-\frac{3}{8}\mathrm{e}^{-2t}\mathrm{sign}\,(t)+\frac{3}{8}$$

在图5-8(b)的传递函数模型后面加一个传递函数模块，并将分子多项式设置为1，分母多项式设置为[T,1]，则可以构造出如图5-10所示的Simulink仿真模型。

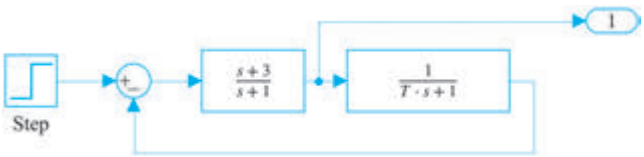


图5-10 人为引入低通滤波器的仿真模型（文件名:c5malg2.slx）

选择滤波常数$T=10^{-6}$，对上面的模型进行仿真，并计算理论值，则可以得出仿真结果与理论值曲线，如图5-11所示。从显示的仿真结果看，二者似乎没有什么区别。

```
>> T=1e-6; t=tout;  %尝试不同滤波参数进行仿真
   y0=exp(-2*t)/8+3*sign(t)/8-3*exp(-2*t).*sign(t)/8+3/8; %理论值计算
   plot(t,[yout y0])          %理论值与数值解曲线比较,在初始时刻有误差
```

事实上，在初始时刻，二者是有显著区别的。由于原始模型直馈现象的存在，输入信号在$t=0$时刻就会在输出信号中体现出来，这时输出信号的初值为0.5。如果采用低通滤波器，则这样的直馈现象被回避了，输出信号在极短时间内($t<10^{-5}$)由0平缓地变化到0.5，后续的响应与理论值很接近。这个过程是很短的，如果T足够小，这个过程是可以忽略不计的。除去这个短暂的时间段，可以得出最大误差为5.0079×10^{-7}，计算点数为1353。

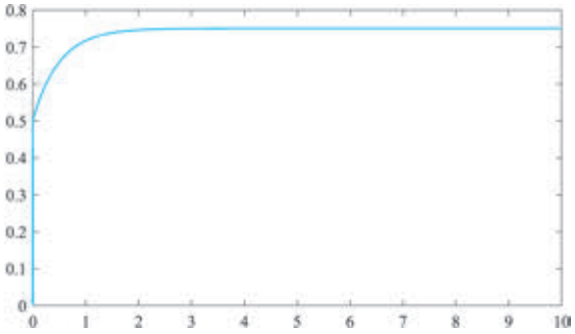


图5-11 时域响应曲线的比较

```
>> ii=t>1e-5; max(abs(y0(ii)-yout(ii))), length(t)
```

减小T值，则可以得出不同T值的仿真结果对比（见表5-1）。可以看出，不加低通滤波器（即表中的“直馈”）虽然会引发代数环现象，耗时稍长，但其仿真结果是最好的。一般情况下不必理会代数环，让Simulink自行求解代数方程即可。在某些特定的场合下，Simulink求解出现困难时，可以考虑引入低通滤波器消除代数环。可以看出，如果T足够小，逼近的效果还是比较好的。

表5-1 不同T值的仿真结果对比

滤波常数T	直 馈	10^{-5}	10^{-6}	10^{-8}	10^{-10}
最大误差	1.6875×10^{-14}	0.0671	5.0079×10^{-7}	4.9999×10^{-9}	4.9999×10^{-11}
计算点数	823	1501	1353	1219	1128
耗时/s	0.5425	0.1911	0.2070	0.1707	0.1607

5.1.4 过零点检测

过零点（zero-crossing）在系统仿真与很多其他领域是一个很重要的概念。考虑一个信号$u(t)$，如果在第k步仿真时刻t_k，$u(t_k)>0$，而下一步t_{k+1}时，$u(t_{k+1})<0$，则在(t_k,t_{k+1})时间段内，$u(t)$信号至少完成了一次零值的穿越，穿越点ξ就是过零点。如果想实现精确的仿真，必须将过零点ξ找出来，否则可能带来难以避免的仿真误差。现在给出一个演示过零点的例子，并介绍检测过零点必要性与代价。

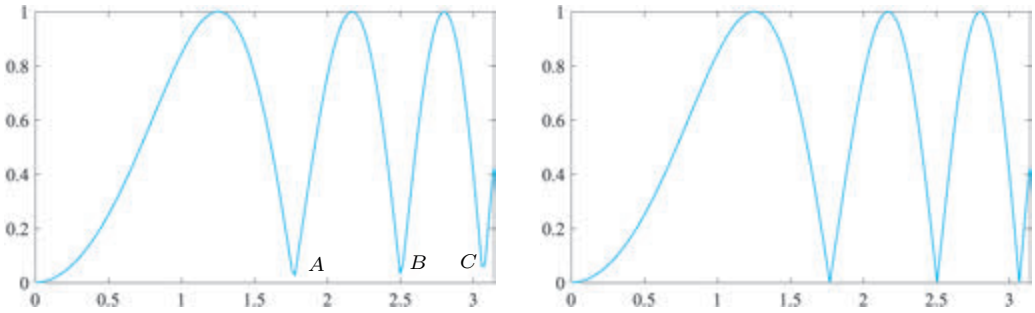
例5-6 试绘制函数$y=|\sin t^2|$的曲线，$t\in(0,\pi)$。

解 如果选择计算步长$T=0.02$s，则可以由下面的MATLAB语句绘制函数的曲线，得出的曲线如图5-12(a)所示。

```
>> t=0:0.02:pi; y=abs(sin(t.^2)); plot(t,y)
```

显然，曲线上有些点处的值是错误的。例如，函数值接近0的几个点（A、B、C）处，函数值应该先下降到0值，再逐渐上升，而不是在悬空的A点进行转换。若能先通过求解方程将函数值等于0的点求出来，再将该点添加到向量t中，问题就能圆满解决。寻找A点的方法就是前面所说的过零点检测。

如果有一种机制能够找出过零点，例如，若使用下面的语句，利用前面给出的解方程方法找到过零点，再将其插入时间向量，则可以得出如图5-12(b)所示的曲线。可以看出，该曲线比较好地处理了过零点，所以该曲线是正确的。



(a) 不检测过零点　　(b) 检测过零点

图5-12　$y=|\sin t^2|$曲线及过零点演示

```
>> f=@(x)abs(sin(x^2)); multisolve(f,zeros(1,1,0),[0,pi])
   t0=X(:); t0=t0(t0>0 & t0<pi);                    %寻找感兴趣范围内的过零点
   t=sort([t,t0']); y=abs(sin(t.^2)); plot(t,y)     %检测过零点并绘图
```

在变步长的微分方程求解算法中，若误差限设置得足够小，则从效果上会自动检测过零点；而在定步长算法中，并不能真正保证检测过零点，所以数值仿真结果难免出现类似图5-12(a)的现象。由此可见，变步长算法与较小的相对误差限是精确求解仿真问题的必备条件。

默认条件下Simulink绝大部分模块是会自动进行过零点检测的，准确检测出过零点是保证精确仿真的有利条件。从仿真性能的角度看，用户不应该对默认的过零点检测选项做更改。不过开启过零点检测的代价是仿真速度减慢，因为在过零点附近，仿真系统会自动求解代数方程。若对仿真精度要求不高，只想做大致的近似仿真，则可以取消过零点检测，加快仿真进程。

5.1.5　仿真过程的快速重启

不论用界面启动仿真进程还是用命令式启动仿真，用户会发现在仿真之前，Simulink模型的状态栏都提示“在编译”。这是正常现象，默认设置下每次仿真都需要重新编译模型。如果解决某问题需要连续多次启用仿真进程，若每次都重新编译，整个过程是极其耗时的。如何避免重新编译呢？Simulink提供了“快速重启”(fast restart)的仿真模式。只要模型结构和不可调参数不发生变化，只是可调参数发生变化，则无须重新编译，将可调参数重新赋值后就可以直接仿真。可调参数的赋值一般应该在模型的工作空间(model workspace)完成，而不是在MATLAB工作空间实现。

快速重启可以单击Simulink工具栏的“快速重启”( )按钮实现，该按钮是双态按钮，再单击该按钮将结束快速重启状态。

对一般仿真而言，不必太关注快速重启问题，可以设置快速重启，也可以不设置。不过在需要大量重复调用仿真过程时，例如，在最优化过程中嵌入Simulink仿真过程时，则必须设置快速重启模式，这样可以大大加速最优化过程，后面将给出快速重启的例子。

5.2　非线性环节与查表环节构建

Simulink中提供了各种各样的非线性模块，使得非线性系统的仿真变得很容易。本节将通过几个例子来演示各种非线性环节的使用与拓展，试图给出通用的非线性环节建模方法。

5.2.1 单值非线性模块

从Simulink中提供的非线性模块组看可能会觉得，该模块组只提供了有限几种静态非线性模块，不一定能包含所需要的所有模块。事实上，很多非线性模块可以由该模块组中给出的模块搭建而成。例如，既带有死区又带有饱和的非线性环节可以由一个死区非线性环节和一个饱和非线性环节串联而成，如图5-13所示[2]。

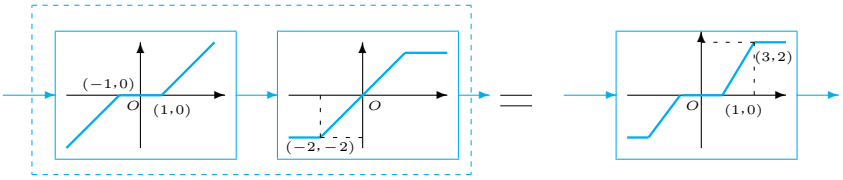
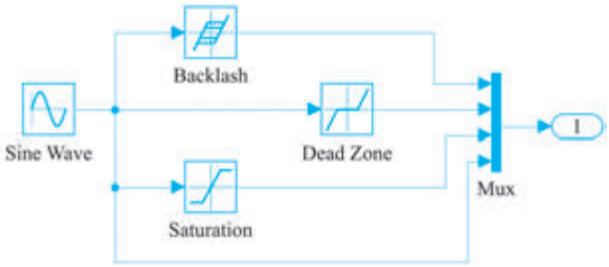


图5-13 复杂非线性环节的等效搭建

例5-7 在正弦信号激励下，经过几个常用的非线性模块，如磁滞回环、死区特性和饱和特性等，试用仿真方法观察输出信号的曲线形状。

解 出于这个目的，可以很容易地构造出如图5-14(a)所示的系统框图，再设置正弦信号的“振幅”为1，并将各个非线性环节的参数设置为c1，如图5-14(b)所示。首先设置c1=0.5，对整个系统进行仿真将得出tout和yout两个变量，用下面的命令可以绘制出正弦信号经过非线性环节后的结果，如图5-15所示。



(a) 仿真框图(文件名:c5nlsin.slx)

(b) Backlash模块参数设置

图5-14 非线性模块仿真框图

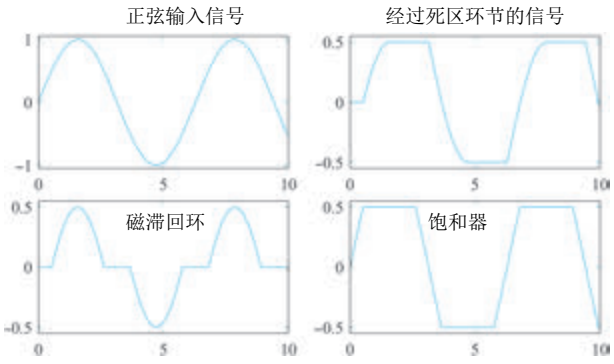


图5-15 $c_1 = 0.5$时的输出比较

```
>> t=tout; y=yout;
   subplot(221), plot(t,y(:,4)), subplot(222), plot(t,y(:,1))
   subplot(223), plot(t,y(:,2)), subplot(224), plot(t,y(:,3))
```

下面探讨一般单值非线性函数的建模方法。单值非线性函数是指输入信号值$u(t)$在每个t时刻对应唯一输出信号值$y(t)$的函数，其数学表达式为$y(t)=f(u(t))$。

现在考虑分段线性的静态非线性函数模块的建模方法。单值非线性静态模块可以由一维查表模块构造出来。考虑如图5-16所示的分段线性非线性静态特性，已知非线性特性的转折点为$(x_1,y_1),(x_2,y_2),\cdots,(x_{n-1},y_{n-1}),(x_n,y_n)$。如果想用Simulink的查表模块表示此非线性模块，则需要在x_1点之前任意选择一个x_0点，即$x_0<x_1$，这样可以根据非线性函数本身求出该点对应的y_0值。同样还应该任意选择一个x_{n+1}点，使得$x_{n+1}>x_n$，并根据延长线求出y_{n+1}的值，这样就可以构造两个向量xx和yy，使得

xx=[$x_0,x_1,x_2,\cdots,x_n,x_{n+1}$]; yy=[$y_0,y_1,y_2,\cdots,y_n,y_{n+1}$];

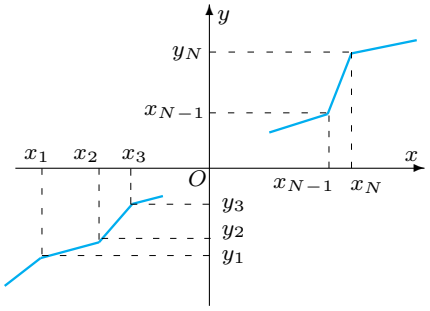


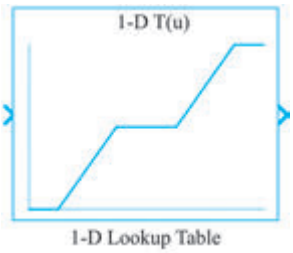
图5-16 分段线性单值非线性模块的一般形式

由Simulink的Lookup Tables模块组提供的1-D Lookup Table模块可以搭建单值的分段线性的非线性环节，下面将通过例子演示模块搭建方法。

例5-8 考虑图5-13中给出的非线性环节，试用Simulink建立该环节的仿真模型。

解 先得出该环节的转折点坐标为$(-4,-2),(-3,-2),(-1,0),(1,0),(3,2),(4,2)$。由于第一段和最后一段都是水平线，所以不用求出扩展点x_0和x_{n+1}，直接引入查表模块，并在图5-17(a)所示“断点1”编辑框中填写[−4,−3,−1,1,3,4]，在“表数据”编辑框中填写[−2,−2,0,0,2,2]，最终由查表模块建立所需的非线性环节，这时该模块图标将自动变成如图5-17(b)所示的形式。

(a) 一维查表模块对话框

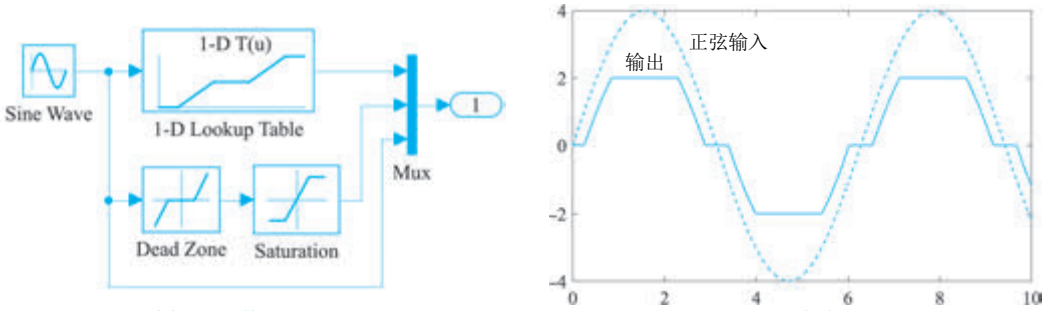


(b) 查表模块图标

图5-17 一维查表模块(文件名:c5mtab1.slx)

掌握了这样的方法，就可以依赖查表模块轻易地建立起任意无记忆的非线性环节。可以将这样的模块建立起仿真模型，如图5-18(a)所示。在该图中还绘制了按图5-13方式搭建的非线性模块，其中死区模块的参数为−1和1，饱和非线性模块的参数为−2和2。

假设输入信号的幅值为4，则正弦信号经过该环节后的输出效果如图5-18(b)所示，且两种方式得出的输出完全一致。还同时叠印输入的正弦信号，可见，非线性模块对输出信号产生了畸变。



（a）仿真框图（文件名：c5mtab2.slx）　　（b）仿真结果

图5-18　非线性模块仿真框图

5.2.2　多值非线性记忆模块

由给出的例子可以看出，任何的单值静态非线性函数均可以采取查表模块的方式来建立或近似，但如果非线性中存在回环或多值属性，则这样的方法是不能直接简单地使用的，解决这类问题则需要使用开关模块和记忆模块。

例5-9　试由Simulink构造一个模型，描述如图5-19所示的回环非线性环节。

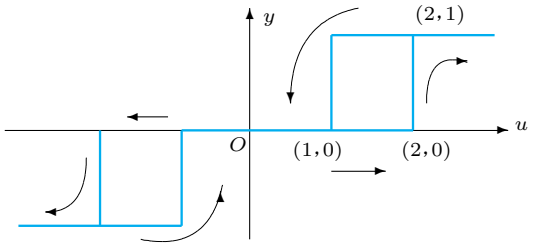
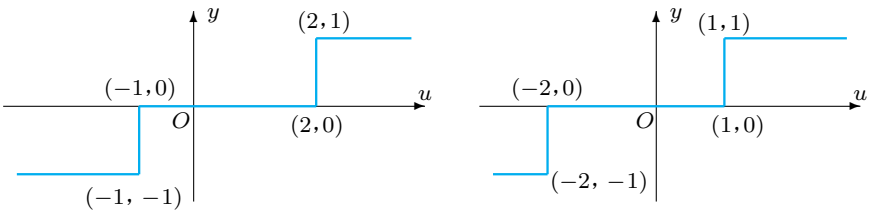


图5-19　给定的回环函数表示

解　可以看出，该特性不是单值的，该模块中输入在增加时走一条折线，减小时走另一条折线。将这个非线性函数分解成如图5-20所示的单值函数，当然这个单值函数是有条件的，它区分输入信号上升还是下降。



（a）当输入量增加时　　（b）当输入量减小时

图5-20　回环函数分解为单值函数

Simulink的离散模块组中提供了一个Memory（记忆）模块，该模块记忆前一个计算步长上的信号值，所以可以按照图5-21中所示的格式构造一个Simulink模型。在该框图中使用了一个比较符号来比较当前的输入信号与上一步输入信号的大小，其输出是逻辑变量，在上升时输出的值为1，下降时输出的值为0。由该信号可以控制后面的开关模块，设开关模块的“阈值”为0.5，则当输入信号为上升时由开关上面的通路计算整个系统的输出，而下降时由下面的通路计算输出。

两个查表模块的输入输出分别为：

$x_1=[-3,\ -1,\ -1+\epsilon,2,2+\epsilon,3],\ y_1=[-1,\ -1,0,0,1,1]$

$x_2=[-3,\ -2,2+\epsilon,1,1+\epsilon,3],\ y_2=[-1,\ -1,0,0,1,1]$

其中,ϵ可以取一个很小的数值,例如,1e-10。设输入正弦信号的幅值为3,则可以得出如图5-22(a)所示的仿真结果,其中虚线表示的仍为输入的正弦信号。

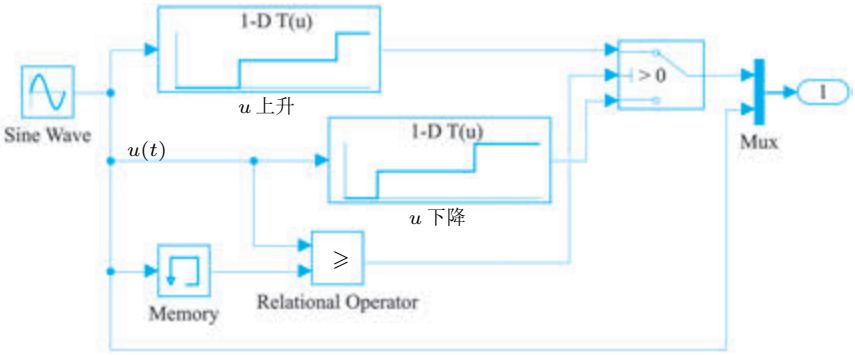
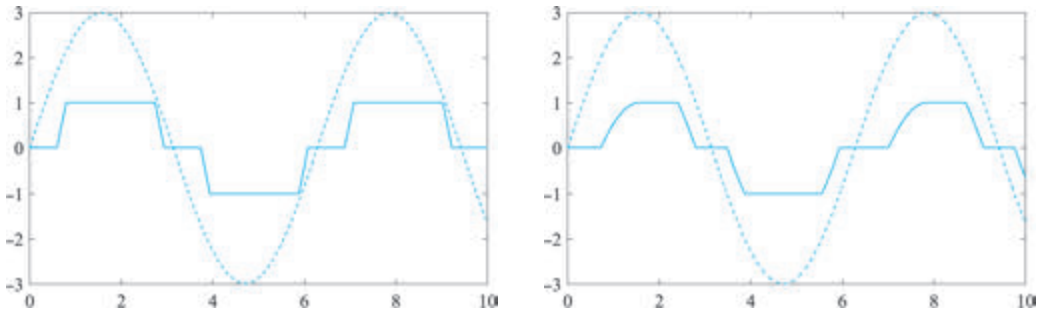


图5-21 回环函数的仿真模型(文件名:c5mloop1.slx)



(a)第一个模型的响应 (b)第二个模型的响应

图5-22 正弦信号在两个回环函数中的仿真结果

例5-10 修改非线性回环函数的结构,使其如图5-23所示,则仍可以利用前面建立的Simulink模型,只需修改两个查表函数为:

$x_1=[-3,\ -2,\ -1,2,3,4],\ y_1=[-1,\ -1,0,0,1,1]$

$x_2=[-4,\ -3,\ -2,1,2,3],\ y_2=[-1,\ -1,0,0,1,1]$

从而立即就能得出整个系统的框图,如图5-24所示。

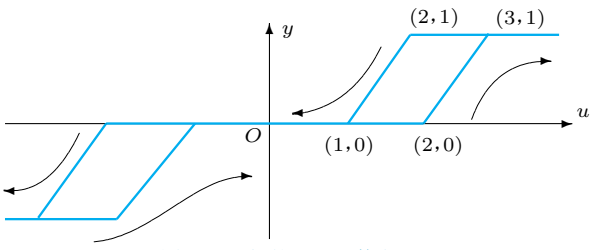


图5-23 新的回环函数表示

对该模型进行仿真分析,则分别得出图5-22(b)中的结果。

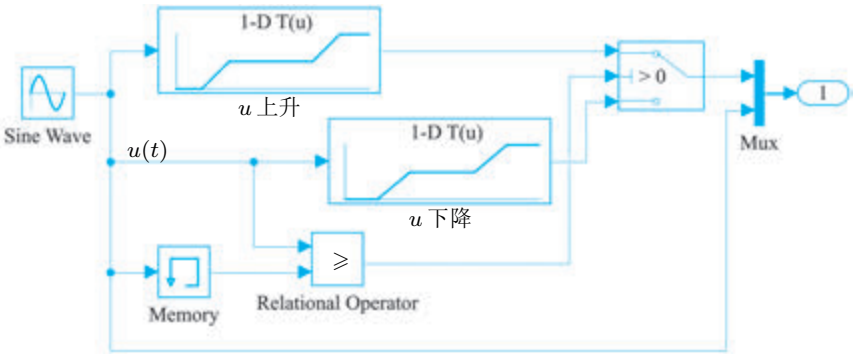


图5-24 修改后的仿真模型（文件名：c5mloop2.slx）

Simulink在信号路由模块组中还提供了Multiport Switch（多路开关）和Manual Switch（手动开关）模块，前者如图5-25（a）所示，最上面的端口为控制信号，该信号等于1，2，…，n时分别连通多路开关对应的输入信号，若不等于这些数值则将给出错误信息。可以双击该模块，在打开的如图5-25（b）所示的对话框中选定多路开关输入信号的路数。

（a）多路开关　　（b）多路开关设置对话框　　（c）手动开关

图5-25 多路开关和手动开关

手动开关也是Simulink的信号路由模块组中很具特色的模块，如图5-25（c）所示，该模块在用户双击时切换状态。在很多场合都可以使用这样的开关来模拟实际对象，如自整定PID控制器的仿真中可以采用手动开关来切换系统的运行状态，从而达到自整定的目的。

例5-11 再考虑例5-7中的问题，可以采用多路开关的方式来选择非线性环节，最终直接返回计算结果，试建立Simulink仿真模型。

解 根据这样的想法，可以绘制出如图5-26所示的多路开关系统的Simulink框图。在该框图中设置了一个外部输入常数key，在MATLAB工作空间中可以定义其值，如key=4。

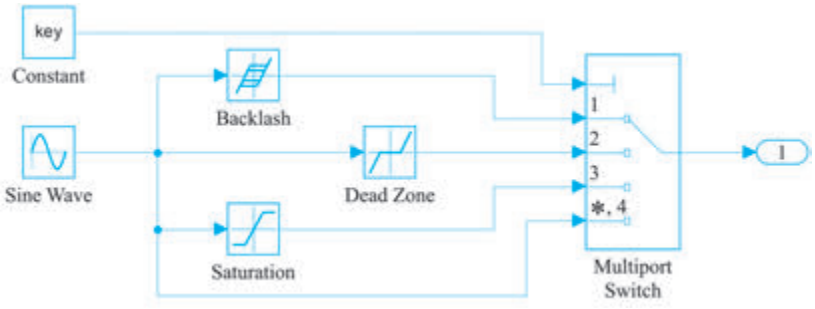


图5-26 多路开关仿真模型（文件名：c5mmswi.slx）

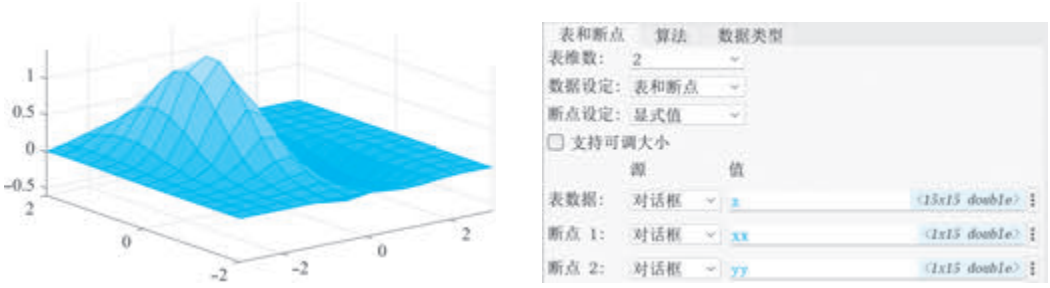
5.2.3 多维查表模块

Simulink不但提供了一维查表的模块，还提供了二维和多维查表模块。提供的多维查表可以使用线性插值的算法求出输出。下面以二维查表模块为例演示多维查表的应用。

例5-12 考虑例2-36中给出的二维函数 $z=f(x,y)=(x^2-2x)\mathrm{e}^{-x^2-y^2-xy}$，假设能构造较稀疏的 xoy 平面上的网格点，并可以计算出在这些点上的高度值：

```
>> xx=linspace(-3,3,15); yy=linspace(-2,2,15);
   [x,y]=meshgrid(xx,yy); z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);
   surf(xx,yy,z); axis([-3,3,-2,2,-0.6,1.4])
```

这样已知点的数据就可以用xx、yy和z表示出来，如图5-27(a)所示，试建立仿真模型。



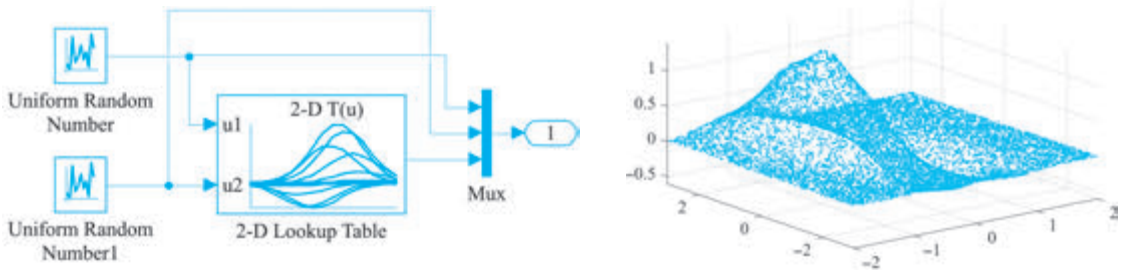
(a) 现有数据表面图　　(b) 二维查表参数对话框

图5-27 多维查表模块使用

解 仿真模型的核心是二维查表模块。双击二维查表模块图标打开如图5-27(b)所示的对话框，在其中可以对描述样本点的3个变量分别进行相应的设置。

现在试图用均匀分布随机数的方式生成一组 $\boldsymbol{x}$ 和一组 $\boldsymbol{y}$ 变量作为输入信号，输入到二维查表模块，就可以构造出如图5-28(a)所示的Simulink框图。注意，在两个随机数发生模块中，伪随机数的种子不能选择相同的值，否则两路输入信号将是一致的，不能很好演示这里的例子。例如，可以将第二路随机数种子选择为1000。另外，假设两个随机数发生模块的范围分别为 $(-3,3)$ 和 $(-2,2)$。在仿真中如果选择定步长为0.001 s，并设定终止仿真时间为10，则可以用仿真的方法计算出10000个点，利用这些点用描点的方式就可以绘制出三维图，如图5-28(b)所示。

```
>> plot3(yout(:,2),yout(:,1),yout(:,3),'.'); axis([-2,2,-3,3,-0.6,1.2])
```



(a) Simulink框图(文件名:c5mtab2d2.slx)　　(b) 三维图形

图5-28 二维查表模块设置与仿真

可以看出其形状类似于已知数据得出的图形。遗憾的是，这个模块采用的是线性插值的方法，所以在精度上较差，一般不能得到平滑的效果。其实，打开查表模块的参数对话框，在其“算法”选项卡

下，可以选择其他插值方法计算输出，例如选择其中的“三次样条”选项，则可以实现输出信号的三次样条插值计算。

Simulink还允许进行多维查表运算，输入输出数据给定还是比较麻烦的，因为它要求给出网格数据，而实测的样本点数据如果不是由网格型给出的，则可以通过griddata()函数对其进行预处理，获得网格数据，再利用查表模块来构造插值模型。

5.2.4 静态非线性模块的代码实现

所谓静态非线性函数就是可以由$\boldsymbol{y}=\boldsymbol{f}(\boldsymbol{u})$函数能直接描述的函数，其中，$\boldsymbol{u}$为模块的输入信号，$\boldsymbol{y}$为模块的输出信号，它们均可能是向量型信号。静态非线性函数的输出信号可以由输入信号直接计算出来。前面介绍过，Simulink自带的Fcn模块不支持向量输出形式，所以有时建模比较麻烦。除了标准的Fcn模块外，Simulink还支持用户自定义的MATLAB函数模块，更简单地，还支持嵌入式MATLAB函数模块。利用这两个模块可以很好地描述静态非线性模块。下面将介绍这两个模块的使用方法。

（1）解释性MATLAB函数模块。用户可以将Interpreted MATLAB Function模块复制到框图中所需位置，然后编辑一个.m文件，将此文件名通过对话框输入到该模块中即可。

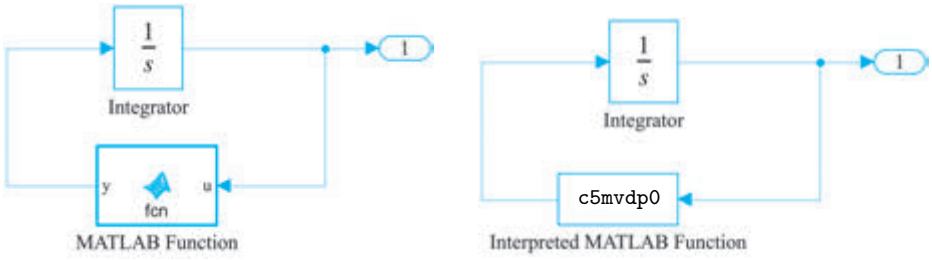
（2）嵌入式MATLAB函数模块。前面介绍的解释性MATLAB函数模块需要在工作路径下建立一个.m文件，而这样做对很多用户来说不大方便，所以可以考虑采用MATLAB Function（嵌入式MATLAB函数）模块来实现。用户只需将该模块复制到所需位置，双击该模块可以自动打开编辑器窗口，允许用户编写M-函数。这样做的好处在于，编写的嵌入式MATLAB代码无须用户单独存成文件，但这样的方式也有劣势，它要求输入与输出信号的维度是不同的。

例5-13 重新完成例5-1中介绍的van der Pol方程的Simulink建模问题。

解 在该例子中采用了Fcn模块，而该模块不支持向量型信号的处理，所以建模结构比较复杂，且需要Mux和Demux等模块的支持。

参见图4-13中给出的User-defined Functions模块组，Simulink提供了两个支持向量输出的静态函数模块，MATLAB Function模块和Interpreted MATLAB Function模块。这两个模块的特点参见上面的叙述。因此，可以建立如图5-29(a)所示的向量化模型，其中，双击MATLAB Function模块，可以打开编辑器，可以填写如下M-函数：

```
function y=fcn(u)
   mu0=1; y=[u(2); -mu0*(u(1)^2-u(2))*u(1)-u(2)];
end
```



（a）向量化模型1（c5mvdp1b.slx） （b）向量化模型2（c5mvdp1c.slx）

图5-29 van der Pol方程的向量化模型

在函数中，主要描述微分方程右侧的向量，其输入信号名需要由向量x改成u。注意，附加参数mu0必须写入函数，不能调用MATLAB工作空间中的变量。如果使用Interpreted MATLAB Function模块，则可以搭建如图5-29(b)所示的仿真模型。可以编写如下的M-函数：

```
function y=c5mvdp0(u)
   mu0=1; y=[u(2); -mu0*(u(1)^2-u(2))*u(1)-u(2)];
end
```

双击Interpreted MATLAB Function模块，打开参数对话框，填写函数名c5mvdp0，则可以建立模块与M-函数之间的关联，从而对微分方程进行直接仿真，得出完全一致的结果。

值得指出的是，这两种模块只能描述静态非线性环节，即输出信号由输入信号的非线性函数直接计算出来的模块。如果需要动态计算，则应该采用S-函数。S-函数还允许使用附加变量，使用起来将更方便。后面将专门介绍S-函数的模块编写方法及其应用。

5.3 微分方程的Simulink框图求解

微分方程是连续动态系统的数学基础，是系统仿真中必须解决的问题。第3章曾经系统介绍过基于MATLAB语言的微分方程数值解方法，这些微分方程还可以通过基于框图的求解方法直接解出。本节将介绍一般微分方程的框图搭建技巧，然后介绍其他类型微分方程的建模与求解方法，包括微分代数方程、延迟微分方程和分数阶微分方程，其中有些方程用现有的MATLAB函数无法求解，但利用Simulink框图法则可以轻而易举地建模与求解。

5.3.1 一般微分方程的Simulink建模技巧

微分方程的Simulink建模是有规律可循的。在实际微分方程建模时，通常进行如下操作绘制框图。

（1）如果想定义$x(t)$和$\dot{x}(t)$两个信号，则可以引入一个积分器模块，并人为地令其输出端为$x(t)$信号，则其输入端自然为$\dot{x}(t)$信号。用这样的方法可以定义出微分方程建模必备的关键信号。该信号的初值$x(t_0)$可以填写到该模块的参数对话框中。

（2）如果两个信号相等，则可以将两个信号直接连接起来。在建模过程中通常采用这种方法闭合仿真回路。

（3）假设已知信号$u(t)$，则可以在后面连接一个Gain模块，并双击该模块输入放大倍数k，这样就可以构造出$ku(t)$信号。

（4）如果要对信号$v(t)$做非线性算术运算，例如，计算$(v(t)+v^3(t)/3)\sin v(t)$，则可以将信号$v(t)$输入Fcn模块。双击该模块，在弹出的参数对话框中填写`(u+u^3/3)*sin(u)`，该模块的输出信号就是期望的非线性函数。注意，不论输入信号是什么信号，输入Fcn模块参数对话框时只能记作u。如果需要获得向量型输出信号，则需使用MATLAB Function模块。

（5）若干路信号可以馈入Mux模块，模块输出信号就是由这些单路信号构造出的向量型信号；如果向量型信号输入Demux模块，则可以分解为标量型信号或多路向量型信号。

（6）如果想由输入信号$u(t)$构造出延迟信号$u(t-d)$，即$u(t)$信号d's之前的值，可以将信号$u(t)$馈入Transport Delay模块，并双击该模块，在参数对话框中输入参数d，模块的输出信号就是期望的延迟信号。

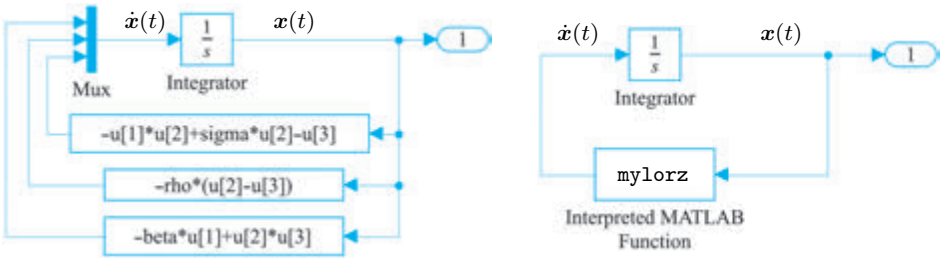
（7）如果想获得某个信号，则可以将其连到Scope模块上，或连接到Out模块上，以便显示或处理该信号。

有了上述的建模规则，则可以很容易地建立微分方程组的仿真模型，无须对微分方程做预处理，构造标准型，直接由绘图的方法将微分方程组用Simulink的模块搭建出来，然后利用Simulink的仿真功能，得出微分方程的数值解。下面将通过例子演示各类微分方程的Simulink建模与求解方法。

例5-14 试建立Simulink仿真模型，求解下面的Lorenz微分方程。

$$\begin{cases} \dot{x}_1(t) = -\beta x_1(t) + x_2(t)x_3(t) \\ \dot{x}_2(t) = -\rho x_2(t) + \rho x_3(t) \\ \dot{x}_3(t) = -x_1(t)x_2(t) + \sigma x_2(t) - x_3(t) \end{cases} \quad \text{且} \quad \begin{cases} \beta = 8/3, \rho = 10, \sigma = 28 \\ x_1(0) = x_2(0) = 0, x_3(0) = 10^{-10} \end{cases}$$

解 用Simulink也可以描述出微分方程组的模型。具体的方法是：考虑原方程中有状态变量向量的一阶导数项，需要使用一个向量型积分器，用其输入端描述$\dot{\boldsymbol{x}}(t)$，其输出端为$\boldsymbol{x}(t)$，其中$\boldsymbol{x}(t) = [x_1(t), x_2(t), x_3(t)]^{\mathrm{T}}$，这样就建立如图5-30（a）所示的Simulink模型。该模型可以直接使用MATLAB工作空间的beta、rho和sigma变量。双击积分器模块，可以将状态变量向量初值[0;0;1e-10]填入积分器模块。



（a）Simulink模型（文件名:c5mlor1a.slx） （b）简单模型（c5mlor1b.slx）

图5-30 Lorenz方程的Simulink建模

遗憾的是，Fcn模块只能处理标量型输出信号，故需要构造如图5-30(b)所示的Simulink模型。在MATLAB函数模块中是不能访问MATLAB工作空间变量的，应该在MATLAB函数中给出赋值语句，建立如图5-30(b)所示的Simulink模型，并关联下面的M-函数：

```
function y=mylorz(x)
   beta=8/3; rho=10; sigma=28; %必须先赋值变量
   y=[-beta*x(1)+x(2)*x(3); rho*(x(3)-x(2)); -x(1)*x(2)+sigma*x(2)-x(3)];
end
```

由类似方法构造的Simulink模型，其结果与ode45()函数的结果完全一致。和前面介绍的ode45()求解函数相比，这里的建模与求解复杂性与该函数求解相仿。

应该指出，对于小规模问题来说，用Simulink建模的求解方式与用ode45()等函数的调用方式相比复杂度相似。但在解决大规模问题、框图化问题以及复杂混联系统的问题时，用Simulink建模方式应该比ode45()类函数调用更方便、直观。此外，对更复杂的时间延迟微分方程求解问题来说，Simulink建模可以解决用普通微分方程求解函数解决不了的问题。

5.3.2 微分代数方程的Simulink建模与求解

在3.4.5节中介绍过微分代数方程的概念，并对给出的例子进行了MATLAB求解，其实用Simulink也能构造出微分代数方程的仿真模型，因为在Simulink的数学函数模块组中给出了代数约束模块，则可以用该模块构造代数方程。

例5-15 试求解例3-34中给出的微分代数方程模型，为方便起见，在这里重新写出原始问题：

$$\begin{cases} \dot{x}_1(t) = -0.2x_1(t) + x_2(t)x_3(t) + 0.3x_1(t)x_2(t) \\ \dot{x}_2(t) = 2x_1(t)x_2(t) - 5x_2(t)x_3(t) - 2x_2^2(t) \\ x_1(t) + x_2(t) + x_3(t) - 1 = 0 \end{cases}$$

并已知初始条件为 $x_1(0) = 0.8, x_2(0) = x_3(0) = 0.1$。

解 因为两个信号 $x_1(t)$ 和 $x_2(t)$ 可以设定为积分器的输出向量，该信号为已知信号，故代数方程为 $f(x_3) = x_1 + x_2 + x_3 - 1 = 0$，用 Algebraic Constraints(代数约束)模块构造出此代数方程的模型表示，其输出信号即解出的 $x_3(t)$ 信号。

如果采用前面介绍的解释性MATLAB函数模块，则可得到如图5-31所示的Simulink模型，其中将两个积分器的初值分别设置为0.8和0.1，并将代数约束的初始值设置为0.1。MATLAB函数模块的内容如下：

```
function y=c5fdae(x)
   y=[-0.2*x(1)+x(2)*x(3)+0.3*x(1)*x(2); 2*x(1)*x(2)-5*x(2)*x(3)-2*x(2)^2];
end
```

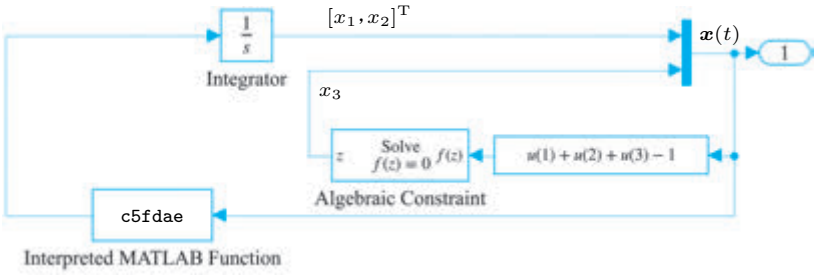


图5-31 微分代数方程的简单仿真框图(文件名:c5mdae.slx)

经仿真可见，该仿真模型可以得出和例3-34完全一致的结果。

5.3.3 延迟微分方程的Simulink求解

延迟微分方程组的一般形式为：

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{x}(t-\tau_1), \boldsymbol{x}(t-\tau_2), \cdots, \boldsymbol{x}(t-\tau_n)) \tag{5-1}$$

其中，$\tau_i > 0$，为状态变量 $\boldsymbol{x}(t)$ 的延迟常数。这样标准的延迟微分方程可以用MATLAB函数dde23()直接求解[3]，但若需要研究中立型延迟微分方程和变延迟微分方程，dde23()则是无能为力的，必须采用基于框图的求解方法。

例5-16 试求下面的零初值延迟微分方程：

$$\begin{cases} \dot{x}(t) = 1 - 3x(t) - y(t-1) - 0.2x^3(t-0.5) - x(t-0.5) \\ \ddot{y}(t) + 3\dot{y}(t) + 2y(t) = 4x(t) \end{cases}$$

解 在第一个方程式中，将$-3x(t)$项移动到等号左侧，则可以将其变换成：

$$\dot{x}(t)+3x(t)=1-y(t-1)-0.2x^3(t-0.5)-x(t-0.5)$$

所以可以将该方程理解成$x(t)$为传递函数模型$1/(s+3)$的输出信号，而该函数的输入信号为$1-y(t-1)-0.2x^3(t-0.5)-x(t-0.5)$。第二个方程式可以理解为：$y(t)$是传递函数模块$4/(s^2+3s+2)$的输出信号，而该模块的输入信号为$x(t)$。在$x(t)$信号和$y(t)$信号上连接延迟模块Transport Delay（传输延迟）可以得出这些信号的延迟。通过上面的分析，可以搭建出如图5-32所示的Simulink仿真模型。对该系统进行仿真则可以得出$y(t)$函数曲线，如图5-33所示。

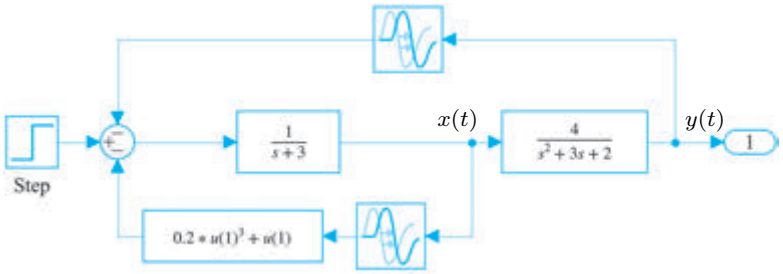


图5-32 延迟微分方程的Simulink模型（文件名：c5mdde1.slx）

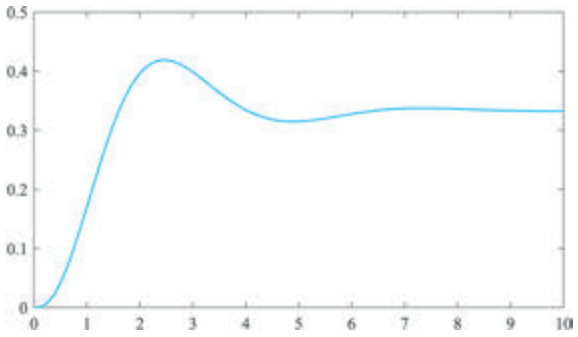


图5-33 延迟微分方程的数值解

当然，若不习惯使用传递函数模块，还可以假设$x_1(t)=x(t),x_2(t)=y(t),x_3(t)=\dot{y}(t)$，这样可以将原微分方程模型变换出如下的一阶状态方程模型：

$$\begin{cases}\dot{x}_1(t)=1-x_1(t)-x_2(t-1)+0.2x_1^3(t-0.5)-x_1(t-0.5)\\ \dot{x}_2(t)=x_3(t)\\ \dot{x}_3(t)=-4x_1(t)-3x_3(t)-2x_2(t)\end{cases}$$

给该状态变量向量选择一个向量型积分器，则可以搭建出Simulink框图，也可以得出同样的结果。这里不给出具体的Simulink模型，读者可以按系统的要求自己搭建该模型。

例5-17 现在考虑中立型延迟微分方程$\dot{\boldsymbol{x}}(t)=\boldsymbol{A}_1\boldsymbol{x}(t-\tau_1)+\boldsymbol{A}_2\dot{\boldsymbol{x}}(t-\tau_2)+\boldsymbol{B}u(t)$，其中，$\tau_1=0.15,\tau_2=0.5$，且

$$\boldsymbol{A}_1=\begin{bmatrix}-13 & 3 & -3\\ 106 & -116 & 62\\ 207 & -207 & 113\end{bmatrix},\ \boldsymbol{A}_2=\begin{bmatrix}0.02 & 0 & 0\\ 0 & 0.03 & 0\\ 0 & 0 & 0.04\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0\\ 1\\ 2\end{bmatrix}$$

试绘制Simulink模型，求解这个中立型延迟微分方程。

解 因为方程中同时包含$\dot{\boldsymbol{x}}(t)$和$\dot{\boldsymbol{x}}(t-\tau_2)$项，这类微分方程又称为中立型(neutral-type)延迟微分方程。该方程用MATLAB自身提供的dde23()函数无法求解，只能采用ddensd()这类专用的函

数求解[3],所以这里考虑采用基于Simulink的框图形式求解该方程。在建模之前,可以用下面的语句输入已知的矩阵:

```
>> A1=[-13,3,-3; 106,-116,62; 207,-207,113];
   A2=diag([0.02,0.03,0.04]); B=[0; 1; 2];
```

再考虑原始的微分方程模型,已经存在一个状态向量$\boldsymbol{x}(t)$,故可以安排一个积分器,使得其输出为$\boldsymbol{x}(t)$,这样其输入端自然是$\dot{\boldsymbol{x}}(t)$,可以分别给这两个信号连接延迟环节,并按实际情况设置延迟时间常数,则可以构造出$\boldsymbol{x}(t-\tau_1)$和$\dot{\boldsymbol{x}}(t-\tau_2)$信号,这样经过简单的处理就可以搭建出如图5-34所示的Simulink模型。该微分方程的解如图5-35所示。

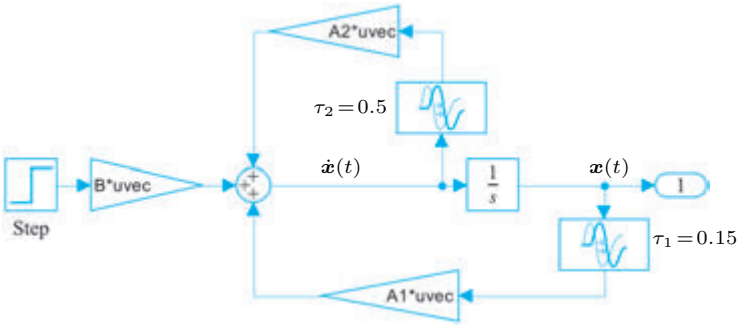


图5-34 中立型延迟微分方程Simulink模型(文件名:c5mdde2.slx)

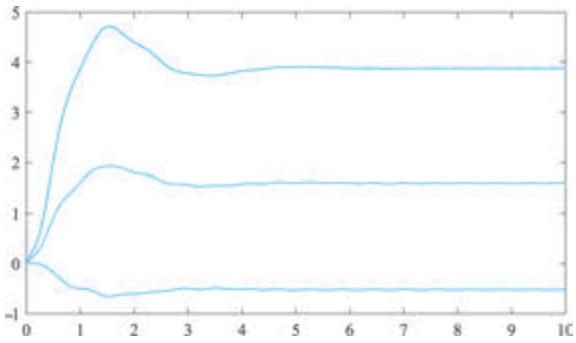


图5-35 中立型延迟微分方程Simulink模型(文件名:c5mdde2.slx)

例5-18 如果各个状态变量初始条件为0,试求解下面的变时间延迟微分方程:

$$\begin{cases} \dot{x}_1(t) = -2x_2(t) - 3x_1(t - |0.2\sin t|) \\ \dot{x}_2(t) = -0.05x_1(t)x_3(t) - 2x_2(t-0.8) \\ \dot{x}_3(t) = 0.3x_1(t)x_2(t)x_3(t) + \cos(x_1(t)x_2(t)) + 2\sin 0.1t^2 \end{cases}$$

解 显然,由于延迟微分方程中存在变时间延迟,即存在$t-0.2|\sin t|$时刻的x_1信号,所以这样的模型用dde23()函数是不能求解的,必须借助于Simulink框图来求解。

和其他微分方程框图建模一样,需要用一个向量化积分器分别定义出$\boldsymbol{x}(t)$信号及其导数信号,这样可以搭建起如图5-36所示的系统仿真框图。注意,在框图中,变延迟时间模型可以调用Variable Time Delay(可变传输延迟)模块,让其第二路输入信号表示变时间延迟$0.2|\sin t|$。为避免悬空信号,在模型中使用了Terminator模块。解释性MATLAB函数模块的M-函数如下。

```
function dx=c5mdde3a(x)
   dx=[-2*x(2)-3*x(5); -0.05*x(1)*x(3)-2*x(4)+2;
```

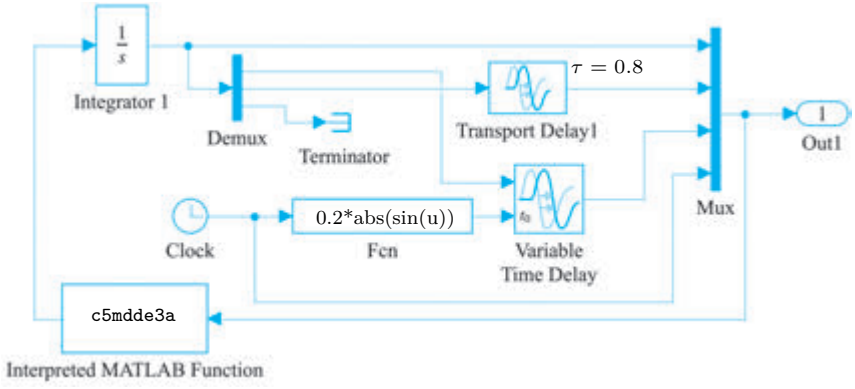


图 5-36 变时间延迟微分方程的 Simulink 模型（文件名：c5mdde3.slx）

```
    0.3*x(1)*x(2)*x(3)+cos(x(1)*x(2))+2*sin(0.1*x(6)^2)];
end
```

对该系统进行仿真，将得出如图 5-37 所示的数值解结果。可以测试不同的仿真控制参数，如相对误差限或仿真算法，以验证结果的正确性。

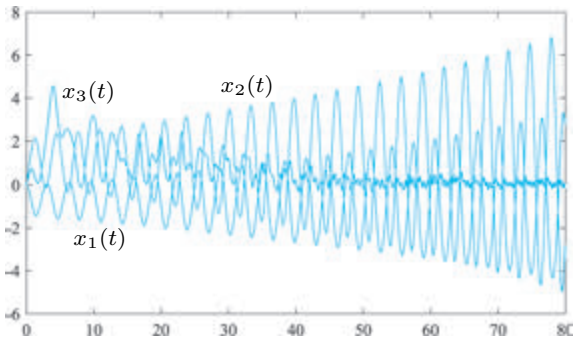


图 5-37 变延迟微分方程数值解

5.3.4 切换微分方程的Simulink求解

切换系统是控制理论中的一个重要的研究领域[4]，就是在某种规律下其模型在多个模型间切换的系统。切换系统的数学模型可以表示为：

$$\dot{\boldsymbol{x}}(t)=\boldsymbol{f}_i(t,\boldsymbol{x},\boldsymbol{u}),\ i=1,2,\cdots,m \tag{5-2}$$

该系统允许在某个控制规律下，整个系统在各个模型之间切换。利用切换系统的理论，可以设计控制器，使不稳定的各个模型 $\boldsymbol{f}_i$ 通过合理的切换达到整个系统的稳定。

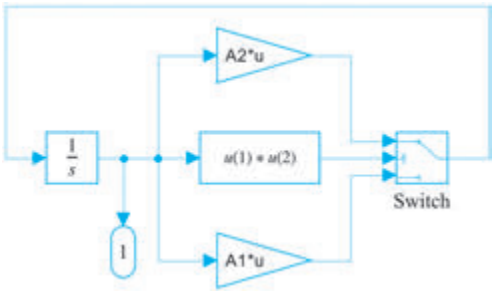
例 5-19 试利用 Simulink 求解下面的多系统模型 $\dot{\boldsymbol{x}}(t)=\boldsymbol{A}_i\boldsymbol{x}(t)$，其中

$$\boldsymbol{A}_1=\begin{bmatrix}0.1 & -1\\ 2 & 0.1\end{bmatrix},\ \boldsymbol{A}_2=\begin{bmatrix}0.1 & -2\\ 1 & 0.1\end{bmatrix},\ \boldsymbol{x}(0)=\begin{bmatrix}5\\ 5\end{bmatrix}$$

解 可见，两个子系统都不稳定。若 $x_1(t)x_2(t)<0$，即状态处于第Ⅱ、Ⅳ象限时，切换到系统 $\boldsymbol{A}_1$；而 $x_1(t)x_2(t)\geqslant 0$，即状态处于第Ⅰ、Ⅲ象限时切换到 $\boldsymbol{A}_2$。

可以用开关模块搭建切换条件，这样可以搭建如图 5-38(a)所示的 Simulink 仿真模型，其中设置开关模块的对话框如图 5-38(b)所示。为实现要求的状态切换条件，需要将开关模块的“阈值”参数设

置为0。此外，为得出精确的仿真结果，需要选中“启用过零检测”复选框，这样就可以利用Simulink的过零点检测功能了。



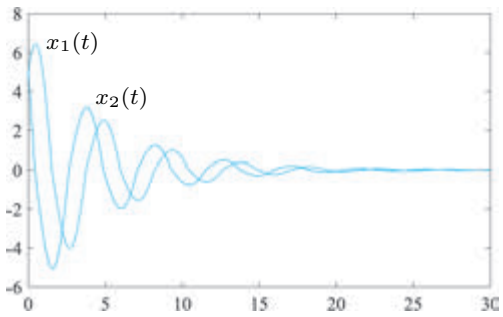
（a）Simulink仿真模型（文件名:c5mswi1.slx）

（b）设置开关模块对话框

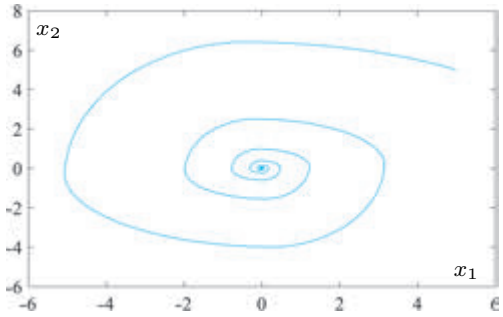
图5-38　切换微分方程的仿真模型

可以将仿真结果返回到MATLAB工作空间，用下面语句可以绘制出状态变量的时间响应曲线和相平面曲线，如图5-39所示。可见，在这里给出的切换条件下，整个系统是稳定的。

```
>> plot(tout,yout), figure; plot(yout(:,1),yout(:,2))
```



（a）状态变量



（b）相平面曲线

图5-39　切换微分方程的解

5.3.5　分数阶微分方程的Simulink求解

分数阶微积分学是传统微积分学的拓展。在分数阶微积分学中，允许微积分的阶次取非整数，所以传统的微分方程仿真算法不能直接使用，一般采用整数阶滤波器，如Oustaloup滤波器[5]来逼近分数阶微分算子s^γ，并可以将该滤波器封装成Simulink模块直接使用。FOTF工具箱提供了FOTF模块集[6]，若安装了FOTF工具箱，则由`fotflib`命令可以打开模块集界面，如图5-40所示，由其中的Riemann-Liouville模块可以逼近s^γ算子，其中，$-1 \leqslant \gamma \leqslant 1$。

例5-20　考虑下面的分数阶非线性微分方程模型：

$$\frac{3\mathscr{D}^{0.9}y(t)}{3+0.2\mathscr{D}^{0.8}y(t)+0.9\mathscr{D}^{0.2}y(t)}+\left|2\mathscr{D}^{0.7}y(t)\right|^{1.5}+\frac{4}{3}y(t)=5\sin(10t)$$

其中，$\mathscr{D}^{0.9}y(t)$表示$y(t)$的0.9阶导数，且初始时刻系统处于静止状态。试建立Simulink仿真模型。

解　如果不考虑初始条件，则可以使用FOTF模块集中的Riemann–Liouville模块直接计算信号的分数阶导数。对原方程稍加变换，则可以写出$y(t)$函数的显式表达式为：

$$y(t)=\frac{3}{4}\left[5\sin(10t)-\frac{3\mathscr{D}^{0.9}y(t)}{3+0.2\mathscr{D}^{0.8}y(t)+0.9\mathscr{D}^{0.2}y(t)}-\left|2\mathscr{D}^{0.7}y(t)\right|^{1.5}\right]$$

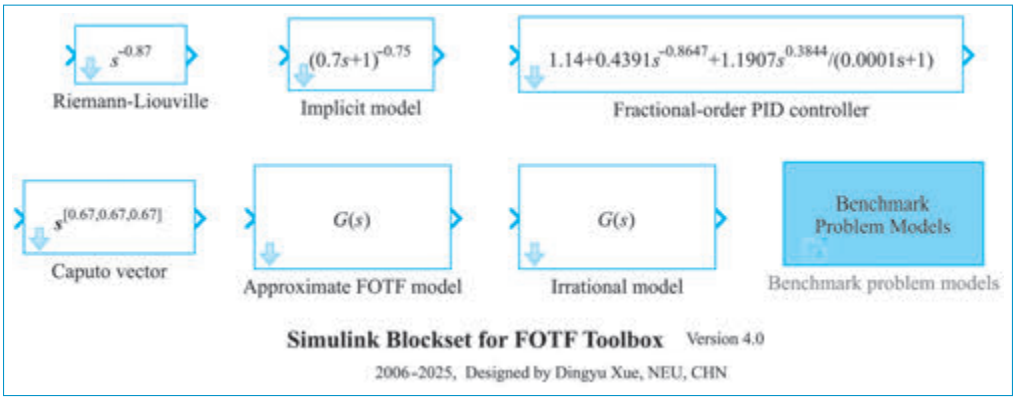


图 5-40 FOTF 模块集

根据得出的 $y(t)$ 可以绘制出如图 5-41 所示的仿真模型。从得出的仿真模型可见，信号的各个分数阶微分信号可以由前面设计的模块获得，因此仿真的精度取决于滤波器对微分的拟合效果，选择不同的拟合频段和滤波器阶次对求解精度将有一定的影响。图 5-42 对不同的滤波器频段、阶次组合进行了比较，得出的结果基本一致，误差稍大的曲线是由 $\omega_b = 0.001\,\mathrm{rad/s}$，$\omega_h = 1000\,\mathrm{rad/s}$，$n = 8$ 得出的。所以对此例来说，选择 $n = 10$ 并选择适当的频段得出的结果几乎完全一致。

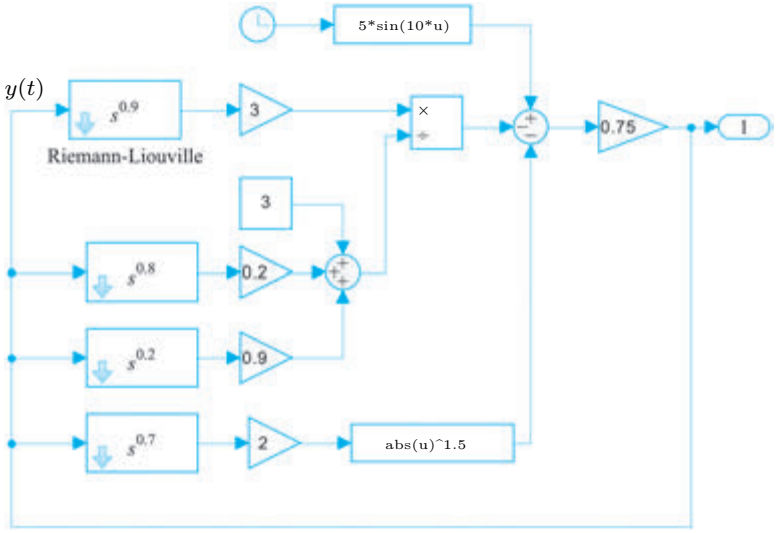


图 5-41 非线性分数阶微分方程的 Simulink 模型（文件名：c5mfode1.slx）

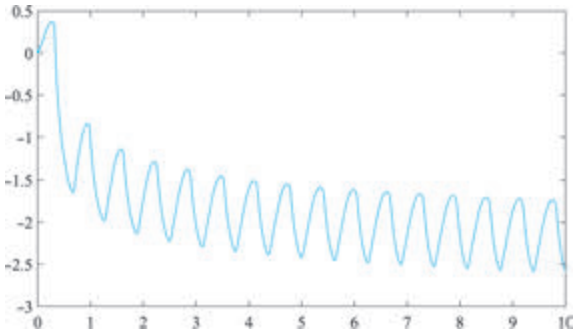


图 5-42 非线性分数阶微分方程的仿真结果

应该注意，同一给定的分数阶微分方程的Simulink建模方法是不唯一的。考虑原始的微分方程，如果将其写成另一种形式：

$$\mathscr{D}_t^{0.9}y(t)=\frac{3+0.2\mathscr{D}_t^{0.8}y(t)+0.9\mathscr{D}_t^{0.2}y(t)}{3}\left[5\sin 10t-\frac{4}{3}y(t)-\left|2\mathscr{D}_t^{0.7}y(t)\right|^{1.5}\right]$$

则新的方程在分母上就不再有动态运算了。从这样的显式微分方程模型可以直接绘制出如图5-43所示的另一个完全不同的Simulink仿真模型，由该模型得出的仿真结果与前面得出的完全一致。

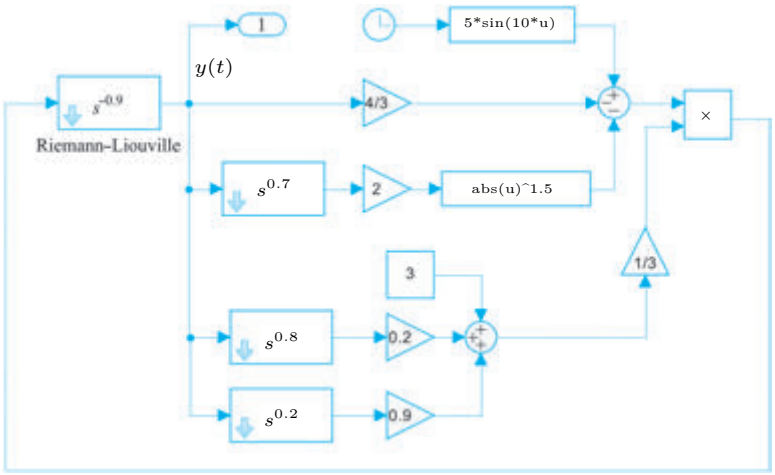


图5-43 非线性分数阶微分方程的另一个Simulink模型（c5mfode2.slx）

对含有复杂运算的微分方程而言，还有一种简洁的建模方法，就是将各个关键信号连同输入信号做成向量型信号，如令$\boldsymbol{v}=[\mathscr{D}_t^{0.9}y(t),\mathscr{D}_t^{0.8}y(t),\mathscr{D}_t^{0.2}y(t),\mathscr{D}_t^{0.7}y(t),u(t)]$，再连接Fcn模块描述复杂运算。这样建立的简洁模型如图5-44所示。启动仿真过程，可以得出与前面模型完全一致的仿真结果。

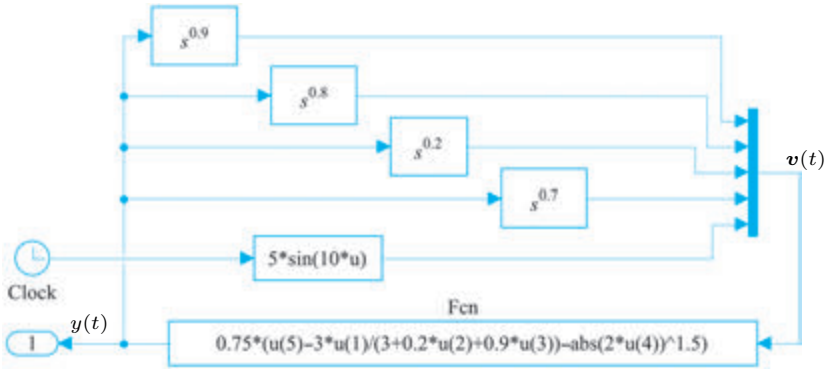


图5-44 非线性分数阶微分方程的简洁建模（c5mnlf3.slx）

若想探讨非零初值的分数阶微分方程，则应该考虑分数阶微积分的Caputo定义[7]。信号的Caputo导数不能利用Riemann–Liouville模块直接计算，而应该采用下面的性质计算。

$$\mathop{\mathrm{C}}_{t_0}\mathscr{D}_t^{\gamma}y(t)=\mathop{\mathrm{RL}}_{t_0}\mathscr{D}_t^{-(n-\gamma)}\left[y^{(n)}(t)\right] \tag{5-3}$$

其中，$n=\lceil\gamma\rceil$。该性质的物理解释为，信号$y(t)$的γ阶Caputo导数可以由整数阶导数$y^{(n)}(t)$经过$(n-\gamma)$阶Riemann–Liouville积分得出，换句话说，由整数阶导数$y^{(n)}(t)$经过Riemann–

Liouville模块得出。例如，若想获得${}_{t_0}^{\mathrm{C}}\mathscr{D}_t^{2.3}y(t)$信号，需要首先获得$\ddot{y}(t)$，将其馈入$-0.7$阶Riemann–Liouville模块，则其输出就是所需的Caputo导数信号。

对式(5-3)两端取$(n-\gamma)$阶Riemann–Liouville导数，则可以得出

$$ {}_{t_0}^{\mathrm{RL}}\mathscr{D}_t^{n-\gamma}\left[{}_{t_0}^{\mathrm{C}}\mathscr{D}_t^{\gamma}y(t)\right]=y^{(n)}(t) \tag{5-4}$$

其物理解释为，对γ阶Caputo导数${}_{t_0}^{\mathrm{C}}\mathscr{D}_t^{\gamma}y(t)$求$n-\gamma$阶Riemann–Liouville导数，则可以得出整数阶导数$y^{(n)}(t)$。换句话说，Caputo导数通过Riemann–Liouville模块则可以得出整数阶导数。例如，若已知${}_{t_0}^{\mathrm{C}}\mathscr{D}_t^{2.3}y(t)$信号，将其馈入0.7阶Riemann–Liouville模块，则该模块的输出就是$\ddot{y}(t)$信号。

基于这两条性质，下面列出Caputo分数阶微分方程求解的一般建模与求解步骤[6]。

(1) **用积分器链定义整数阶微分信号**。如果微分方程中的最高阶次为α，则需要串联$q=\lceil\alpha\rceil$个整数阶积分器，如图5-45所示，这样将构造出所需的输出信号及其各个整数阶导数信号，此外，还可以将已知的初值依次写入各个积分器。

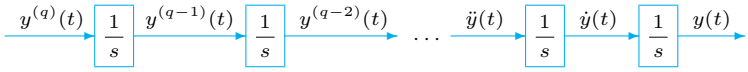


图5-45 整数阶积分器链

(2) **构造所需的分数阶微分信号**。利用式(5-3)中给出的性质，就可以利用Riemann–Liouville模块搭建任意阶次的分数阶Caputo导数信号。

(3) **构造出整个分数阶系统的Simulink仿真模型**。有了所需的整数阶与分数阶导数关键信号，则可以利用Simulink中搭建起整个系统的仿真模型了。在建模过程中，有的时候为了闭合某些回路，可以使用式(5-4)中的性质。建立起来模型后就可以对其仿真，得到原Caputo微分方程的数值解。由于这里介绍的方法是基于框图的方法，所以理论上可以求解任意复杂的Caputo微分方程。

例5-21 试用Simulink求解下面的非线性Caputo微分方程：

$$ {}_0^{\mathrm{C}}\mathscr{D}_t^{1.455}y(t)=-t^{0.1}\frac{\mathrm{E}_{1,1.545}(-t)}{\mathrm{E}_{1,1.445}(-t)}\mathrm{e}^{t}y(t)\,{}_0^{\mathrm{C}}\mathscr{D}_t^{0.555}y(t)+\mathrm{e}^{-2t}-\left[\dot{y}(t)\right]^2$$

其中，$y(0)=1$，$\dot{y}(0)=-1$，已知该方程的解析解为$y(t)=\mathrm{e}^{-t}$。$\mathrm{E}_{\alpha,\beta}(\cdot)$为双参数Mittag-Leffler函数，可以由`ml_func()`函数直接计算。

解 可以依下面的步骤建立Caputo微分方程的仿真模型。

(1) 因为这里的最高微分阶次为1.455，所以$q=2$，需要两个串联的整数阶积分器先定义出$y(t)$、$\dot{y}(t)$与$\ddot{y}(t)$信号。将两个初值分别写入相应的积分器。

(2) 构造关键的${}_0^{\mathrm{C}}\mathscr{D}_t^{0.555}y(t)$信号：由式(5-3)可见，输入信号应引自$\dot{y}(t)$，将其馈入0.445阶Riemann–Liouville模块，该积分器输出就是${}_0^{\mathrm{C}}\mathscr{D}_t^{0.555}y(t)$信号了。有了这些关键信号，就可以由底层模块搭建的方法将方程左侧搭建出来，亦即搭建出${}_0^{\mathrm{C}}\mathscr{D}_t^{1.455}y(t)$信号。

(3) 闭合仿真回路：由式(5-4)可知，如果将其馈入0.545阶Riemann–Liouville模块，则可以计算出$\ddot{y}(t)$，而该信号正巧是积分器链的起点，所以应该将Riemann–Liouville模块得出的信号与$\ddot{y}(t)$直接相连，闭合仿真回路，如图5-46所示。为简单起见，将时间t函数的非线性运算赋给Interpreted MATLAB Fcn模块，其内容为：

```
function y=c5mmlfs(u)     %描述微分方程的M函数
   y=u^0.1*exp(u)*ml_func([1,1.545],-u)./ml_func([1,1.445],-u);
end
```

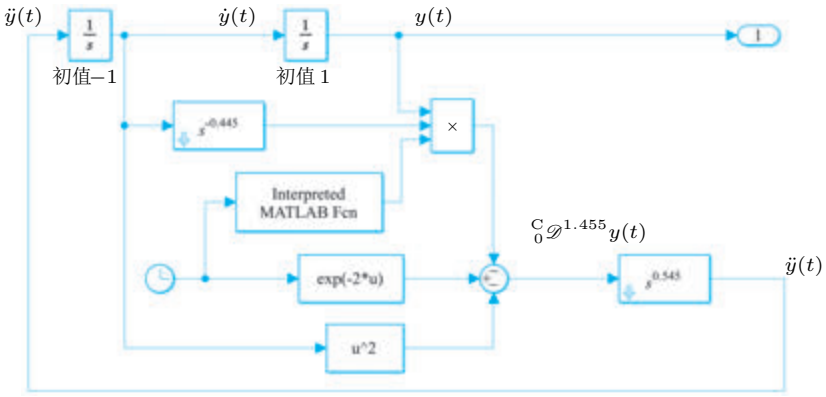


图5-46 一个新的Simulink模型(文件名:c5mexp2s.slx)

为Riemann–Liouville模块选择如下的参数,则可以得出所需的仿真结果,与解析解e^{-t}相比,可以计算出最大的计算误差为4.9058×10^{-5},运行的时间为0.67 s。

```
>> N=18; ww=[1e-7 1e4];      %选择Riemann--Liouville滤波器的参数
   tic, [t,x,y]=sim('c5mexp2s'); toc, max(abs(y-exp(-t)))  %检验
```

如果选择更高阶次并选择更大的频率响应拟合范围,例如,选择频率段$(10^{-7},10^{6})$,并选择阶次$N=30$,则最大误差可以减小到5.9229×10^{-7},所需时间也只需1.84 s。

5.4 输出型模块

输出环节是Simulink仿真中重要的一环,因为仿真的目的是从给定模型得到某种计算结果,而结果是以所需的形式返回给用户的。Simulink允许使用不同的途径处理输出信号。本节主要演示输出信号的常用表现方法和一些相关的内容。

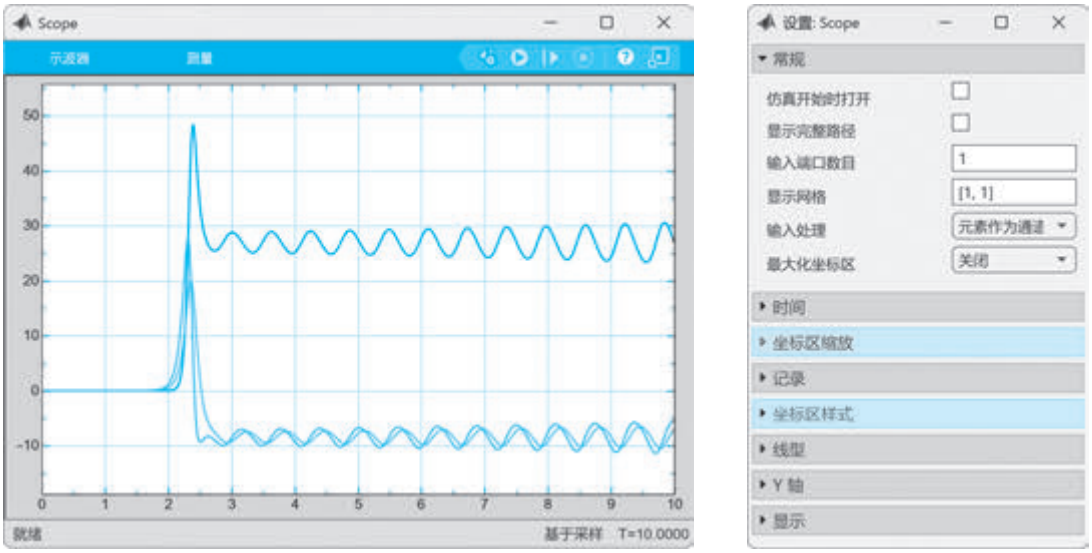
5.4.1 Sinks模块组

4.1.2节介绍了Sinks模块组,其中包括各类示波器模块、直接数据形式模块、工作空间写入模块、文件写入模块、输出端口模块等,可以将感兴趣信号直接连这些模块,观察输出结果,或将仿真结果返回MATLAB工作空间,然后用plot()类函数绘制仿真结果。Sinks模块组还提供了Stop Simulation模块,可以控制仿真进程。如果该模块的输入信号为真,则自动终止仿真进程。

例5-22 考虑例5-14中的c5mlor1b.slx模型,试在其输出端添加一个示波器模块。

解 可以在输出端直接连接Scope(示波器)模块,得到c5mlor2.slx模型。仿真结束后,双击示波器模块,则其显示结果如图5-47(a)所示。单击“示波器 ▸ 设置”按钮,则打开如图5-47(b)所示的参数对话框,允许用户进一步设置示波器参数或显示模态。例如,可以打开“坐标区缩放”扩展框设置坐标轴的状态,或打开“坐标区式样”扩展框设置示波器的背景颜色等。

例5-23 考虑物理学中的物体垂直下抛运动方程$v(t)=v_0+gt$,其中,t为时间,$v(t)$为物体的瞬时速度,$v_0=1$ m/s为初速度,$g=9.81$ m/s^{2}为重力加速度。试构造仿真模型。



（a）示波器显示　　（b）参数设置对话框

图 5-47　控制系统的输出与输入信号

解　已知速度 $v(t)$ 与位移 $x(t)$ 之间的关系为 $v(t)=\dot{x}(t)$，所以原方程可以改写成 $\dot{x}(t)=v_0+gt$。引入一个积分器模块，并假设其输出端为位移 $x(t)$，则其输入端自然就是 $\dot{x}(t)$，这两个信号为关键信号。有了关键信号，就可以搭建如图 5-48 所示的 Simulink 模型。其中，初速度 $v_0=1$ 由 Constant 模块给出，时间 t 由 Clock 模块给出，就构造出可以描述微分方程的 Simulink 模型。

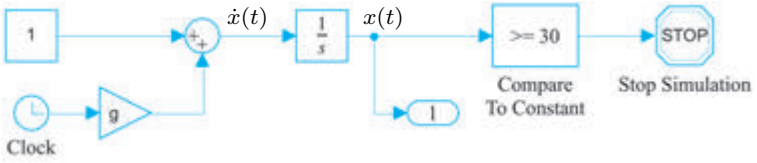


图 5-48　抛体仿真的 Simulink 模型（文件名：c5mmove.slx）

如何设置终止仿真时间呢？假设只已知抛射点离地 30 m，则预先不知道终止时间，无法直接设置。可以考虑在位移 $x(t)$ 达到 30 m 时自动终止仿真进程，这需要在模型中设置 $x(t)\geqslant 30$ 条件，用其驱动 Stop Simulation 模块，强制停止仿真进程。对该系统进行仿真，则由 `tout(end)` 命令可以读出终止仿真时间为 2.3733 s。

5.4.2　信号查看器

在仿真过程中，可以将感兴趣的信号馈入示波器模块，实时地显示该信号的波形。如果同时对很多信号感兴趣，当然可以将每个信号后面都接一个示波器模块，不过这样会使得整个 Simulink 模型布局显得过于凌乱。Simulink 提供了信号查看器，可以利用查看器监测仿真信号，下面通过例子演示查看器的使用方法。

例 5-24　考虑例 5-2 中的模型，试采用查看器观察 $\dot{\boldsymbol{x}}(t)$ 与 $\boldsymbol{Ax}(t)$ 信号。

解　如果对某一路信号感兴趣，可以单击选中该信号线，右击得到快捷菜单，选择“连接到查看器 ▸ Scope”菜单项，则会在该信号线上方标注查看器图标（），也可以选择“创建并连接查看器 ▸ Simulink ▸ Scope”快捷菜单项，在信号线上添加新的查看器。这两种方法的区别是，前者在当前查看

器上再增加选中的信号，后者是新添加查看器模块。利用这样的方法，就可以构造如图5-49所示的仿真模型，其中添加了若干查看器，例如，正弦输入信号、积分器输入端等。仿真时，这些查看器处于关闭状态。仿真结束后，用户可以双击某个查看器图标，显示该路信号。查看器的显示方式类似于示波器。若想删除某个查看器，则先选择该信号线，再选择“删除查看器 ▸ Scope”快捷菜单项实现。

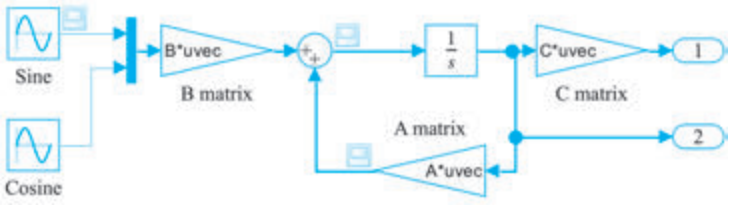


图5-49 添加查看器的Simulink模型(文件名:c5mss1b.slx)

5.4.3 仿真数据检查器

类似于添加查看器的方法，Simulink还允许用仿真数据检查器检测模型中的信号。类似于查看器模块，选中感兴趣的信号线，右击获得快捷菜单，从中选择“记录所选信号”，则该信号出现 图标。若想取消该图标，则从快捷菜单选择“停止记录所选信号”即可。

例5-25 仍考虑例5-2中的模型，试采用仿真数据检查器观察 $\dot{\boldsymbol{x}}(t)$ 与 $\boldsymbol{A}\boldsymbol{x}(t)$ 信号。

解 可以单击选中感兴趣的信号线，右击得到快捷菜单，选择“记录所选信号”菜单项，则会在该信号线上方标注图标 。利用这样的方法，就可以构造如图5-50所示的仿真模型。查看器的界面与示波器完全一致。

图5-50 添加仿真数据检查器的Simulink模型(文件名:c5mss1c.slx)

仿真结束后，单击 图标，则打开如图5-51所示的仿真数据检查器界面，用户可以有选择地在其中显示感兴趣的曲线。仿真数据检查器的右侧底部提供了▶按钮，用户可以单击该按钮回放仿真结果，也可以利用◀|和|▶按钮控制播放速度。

5.4.4 表盘与量计显示

Simulink允许用表盘或量计模块直接显示仿真的结果。打开Simulink的Dashboard模块组，其部分表盘类模块如图5-52所示。每一个表盘模块上方有一个 标识，可以打开参数对话框，与感兴趣的信号进行连接。这样，仿真结果的显示类似于实际过程控制现场见到的显示仪表。下面将通过例子演示表盘模块的使用方法。

例5-26 试为图5-53中给出的控制系统建立Simulink仿真模型，用表盘显示输出信号。

解 用Simulink可以建立仿真框图，将输出信号直接连接Gauge模块，如图5-54所示。具体的Gauge模块连接步骤如下。

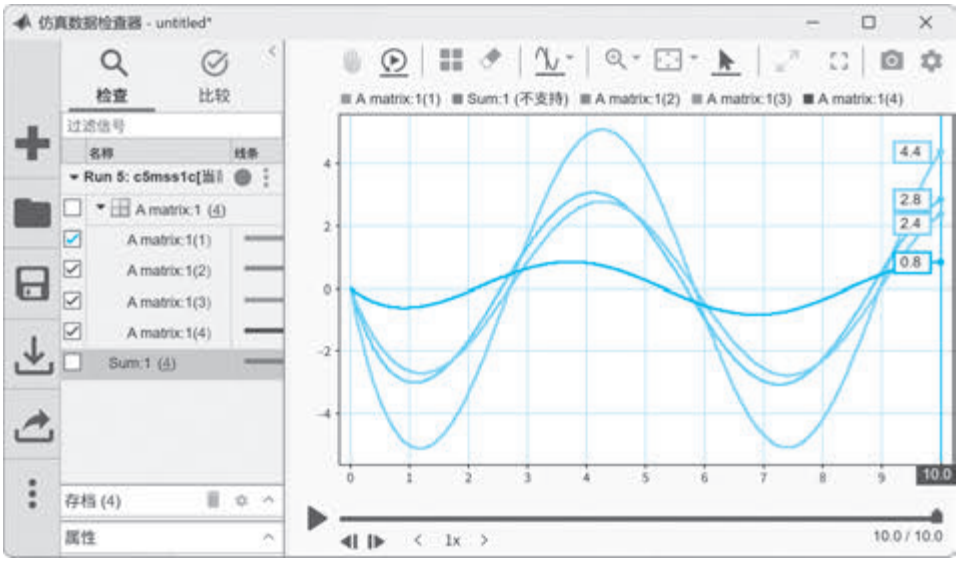


图 5-51 仿真数据检查器显示界面

图 5-52 Dashboard 模块组（部分模块）

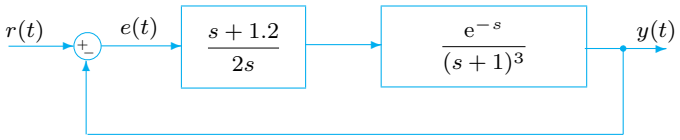


图 5-53 控制系统框图

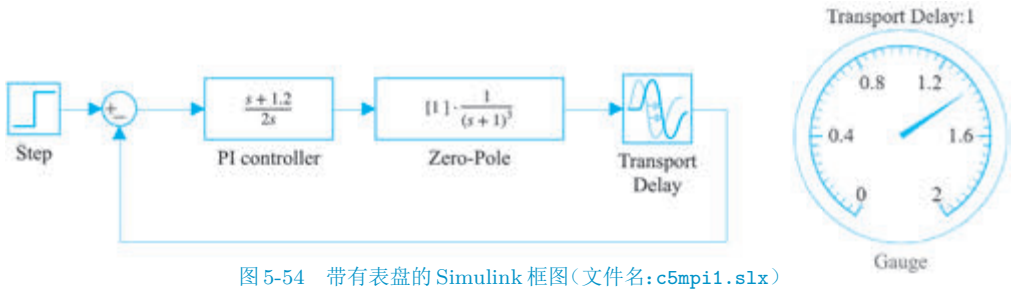


图 5-54 带有表盘的 Simulink 框图（文件名：c5mpi1.slx）

（1）将 Gauge 模块拖动到 Simulink 模型窗口。

（2）双击 Gauge 模块，将显示如图 5-55 所示的对话框。默认情况下，“连接”栏目不连接任何模块。

（3）单击模块的 🔗 标识，则允许用户选择感兴趣的信号线或模块，例如，选择反馈回路线，这时，会自动建立 Gauge 模块与 Transport Delay 模块第一端口的关联，如图 5-55 所示。除了关联关系，还有必要设置表盘的量程，例如，将“最大值”从默认的 100 修改为 2。

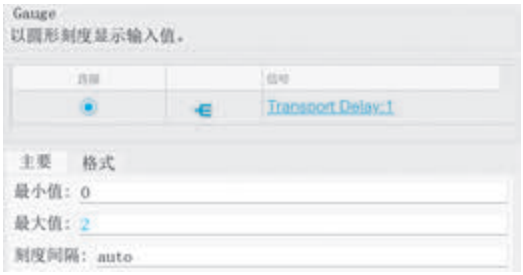


图 5-55　Gauge 模块参数对话框

连接建立之后，在模块上方将显示连接模块的信息(Transport Delay:1)。启动仿真过程，则将以表盘的形式显示系统的输出信号。

值得指出的是，每路输出信号允许同时连接多个输出模块，例如，一个信号既可以连示波器，也可以同时连浮动示波器、输出端口和表盘显示，另外，还可以连接一个工作空间存储数据模块将结果写入工作空间，如图5-56所示。如果连接了 To Workspace 模块，并将其关联为变量 `simout`，则仿真后返回的结果可以由结构体的 `Time` 与 `Data` 成员变量返回。

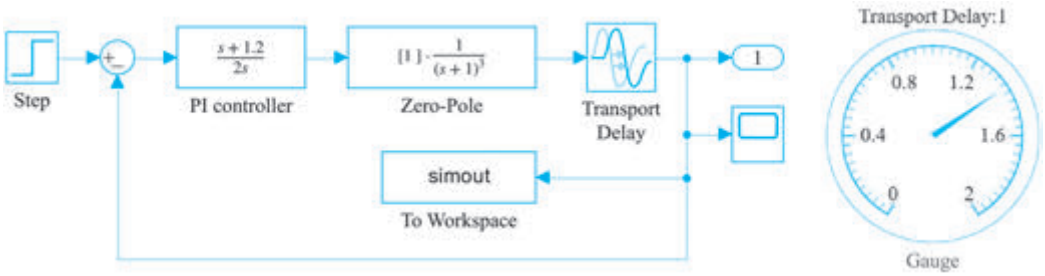


图 5-56　同时接多个输出模块的框图(文件名:`c5mpi2.slx`)

5.4.5　输出信号的数字信号处理

Simulink 在 Simulink Extras(其他模块)组中的 Additional Sinks(附加输出池)提供了一些数字信号处理的模块，如图5-57所示。另外，在 DSP System Toolbox(DSP 系统工具箱)中提供了更强大的模块库，如图5-58所示。在本节中将通过例子来演示数字信号处理的功能。

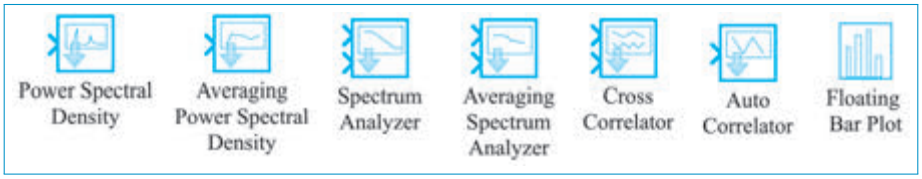


图 5-57　Simulink 附加输出池库

例 5-27　建立 Simulink 模型，分析 $x(t)=12\sin 20t+5\cos 40t$ 信号的自相关函数。

解　可以按照图5-59(a)所示的方式构造出 Simulink 仿真框图，在该框图中，使用了两个正弦输入模块，按照给出函数中的参数分别设置这两个模块的参数。例如，在第二个正弦输入模块的参数对话框中可以按图5-59(b)的形式给出数据，亦即其幅值为5，频率为 40 rad/s，初始相位为 $\pi/2$，这样 $5\sin(40t+\pi/2)=5\cos 40t$ 即为要求的信号。这两个模块叠加之后，将其结果连接到 Auto Correlator(自相关分析器)模块上。

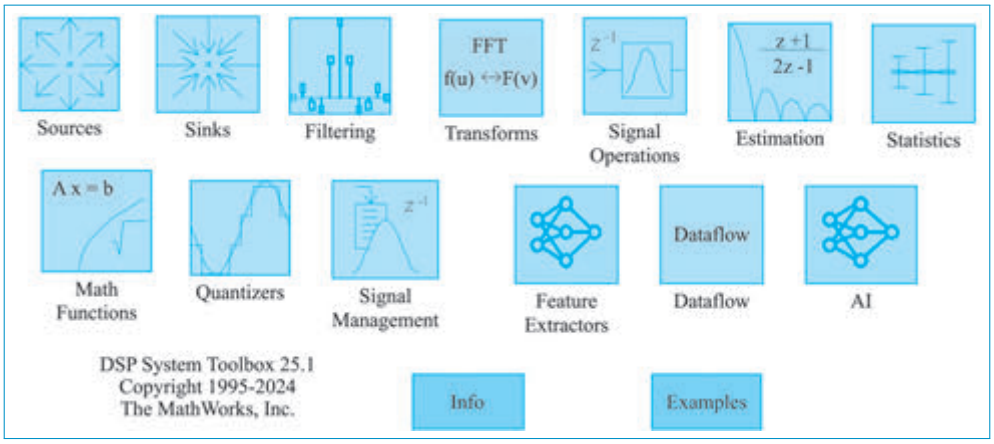
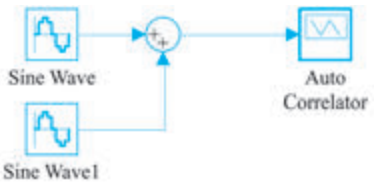


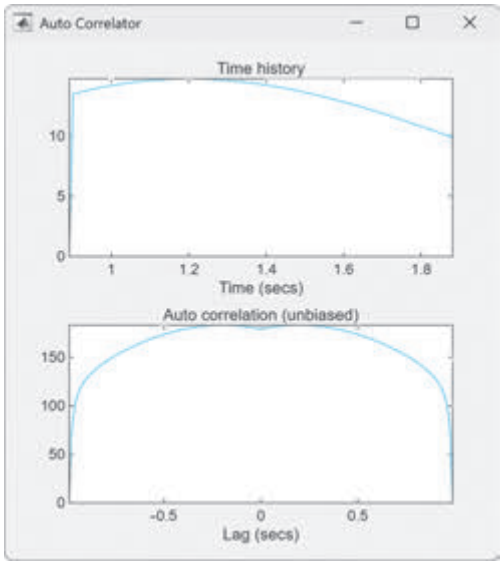
图5-58 Digital Signal Processing模块集

如果想使用自相关模块，则需要将系统仿真设置为定步长的算法，且可以选择步长为$T = 0.01$，这样启动仿真过程，将以滚动的形式显示时间响应曲线和自相关函数，如图5-59(c)所示。



(a)仿真框图(文件名:c5mfft.slx)

(b)余弦输入模块参数



(c)自相关函数显示

图5-59 输入信号的自相关分析

5.4.6 避免交叉连线的方法

复杂系统Simulink模型的连线有时过于复杂，经常出现大量的交叉线，使得整个系统显得很凌乱，给仿真模型的阅读与维护造成了很大的麻烦。例如，在一个大系统模型中，最右侧的某个信号需要馈入左端的一个环节，这就需要建立一条很长的连线，这个连线会贯穿整个模型。如果系统中间部分的连线比较密集，这样引入的长连线很可能与中间的信号线产生交叉。为避免这样的交叉线，可以考虑将右侧的连线连到From模块上，而馈入左边的连线，可以由Goto模块发出，这样就可以不用真正连接一条贯通的连线了。当然，这两个模块需要做一定的匹配，才能确保两个模块真正连接到一起了。这里将通过例子演示这种连线方法。

例5-28 考虑例4-1中给出的问题。想象一下，如果图4-24中$\hat{y}(t)$信号与加法器第二输入端口之间线路比较复杂，不便于直接连线，试利用From模块与Goto模块完成实际连接。

解 打开图4-27中的仿真模型c4mmod1c.slx，并另存为c5mmod1c.slx，删除$\hat{y}(t)$到加法器之间的连线，并将两个悬空的端口分别连接Signal Routing模块组中的Goto和From模块，建立新的仿真模型，如图5-60所示。其中，[A]是标记，如需修改，可以通过参数对话框实现，但两者必须保持一致。

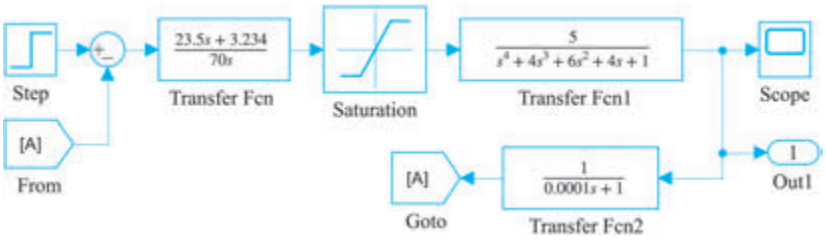


图5-60 修改的仿真模型(文件名:c5mmod1c.slx)

当然，对这样的简单问题而言，没有必要采用这样的间接连线的方法。用户可以采用这样的方法处理更复杂模型的连线问题。

5.5 Simulink仿真结果的三维动画显示

早期版本支持的虚拟现实工具箱已经正式更名为Simulink 3D Animation Blockset(三维动画模块集)，可以由虚拟现实的方法将仿真结果显示出来。本节首先介绍虚拟现实的基本概念，并通过例子演示三维动画建模方法，并演示Simulink 3D Animation模块集的虚拟现实模型驱动与显示方法。

5.5.1 虚拟现实基础

虚拟现实是一种可以创建和体验虚拟世界的计算机系统。虚拟世界是全体虚拟环境或给定仿真对象的全体。虚拟环境是由计算机生成的，通过视、听、触觉等作用于用户，使之产生身临其境的感觉或交互式视景仿真[8]。

虚拟现实必须是一个由计算机所产生的三维立体空间，用户可以和这个空间的对象进行交互，除观看外还可以操作其中部分对象，并可在空间中随用户的意志自由移动，进而产生相对的融入感和参与感[9]。Burdea公司提出了广为认可的3I定义：

(1)**沉浸度**(immersion)。人们全心投入一件事情时，会达到浑然忘我的境界，无视外界的环境。虚拟现实就是借助这种心理让人摆脱现实环境的压力，进入计算机模拟的虚拟现实。

(2)**交互性**(interactive)。真实世界中，人可以和周围的环境交互，所以虚拟现实就是要把这种人与环境间的交互性加入虚拟现实中，让虚拟现实更为真实。

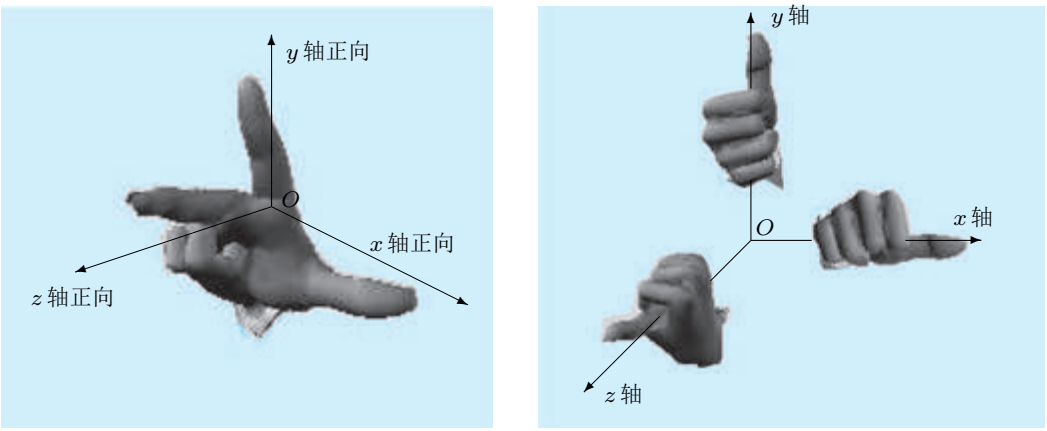
(3)**想象力**(imagination)。虚拟现实就是借助人类的想象力，将虚拟现实和真实的实物联系在一起。

MATLAB的三维动画显示功能允许Simulink使用虚拟现实的显示技术，使用户直接将仿真结果以虚拟现实的形式显示出来。本节将首先介绍虚拟现实的基本概念，再介绍虚拟现实模型语言VRML程序的生成方法，然后介绍三维动画的基本功能，包括在MATLAB环境下和Simulink下仿真结果的虚拟现实显示方法及应用。

VRML（virtual reality modeling language）是一种常用的虚拟现实描述语言，其后续发展的国际标准是X3D（extensible 3D）文件。它继承了VRML的核心功能并采用了基于XML的语法结构，成为更现代、可扩展性更强的3D图形格式。在MATLAB的虚拟现实工具箱中，主要采用这两类语言来描述虚拟现实场景。

VRML/X3D语言在描述三维空间时，其坐标框架满足右手法则，如图5-61（a）所示，即右手的拇指、食指、中指相互垂直，则它们的指向分别构成了x、y、z轴。从该图可见，其坐标轴排列顺序和常规使用的坐标系不完全一致，所以应该注意。

定义了坐标轴方向以后，右手拇指指向坐标轴的正方向，其余四指握拳后所指的方向就是沿该坐标轴旋转的方向，其示意图如图5-61（b）所示。



（a）坐标轴方向右手法则　　（b）坐标轴旋转方向

图5-61　右手法则示意图

5.5.2　Simulink三维动画模型编辑器

MATLAB R2024b及以前版本提供了Simulink三维动画场景建模的编辑器。在MATLAB环境下给出`vredit`命令，则可以打开如图5-62所示的场景编辑界面。该界面分为三个区域，其简单介绍如下。

- **左侧区域**为场景层次面板，当前显示为树状结构，根节点为ROOT。这种树状结构直观展示了三维场景中各个对象的父子关系和层级组织，用户可以在该节点下添加其他的节点。树状结构显示各个节点及其下级属性。
- **下部区域**为注释区域，描述当前节点的注释信息。
- **右侧区域**为三维场景的编辑和可视化区域。用户可以用可视的方式添加与调整各个节点，创建三维动画场景模型。

下面将通过例子演示三维动画场景的创建方法。

例5-29　利用`vredit`程序创建一个三维动画场景的模型，该模型中有一架飞机和一棵大树。

解　用`vredit`命令打开三维动画场景的编辑程序。程序中已经提供了很多预设的节点对象，如本例期望的飞机模型与大树模型。可以由图5-63（a）给出的“Nodes（节点）▸ Insert from（插入）▸

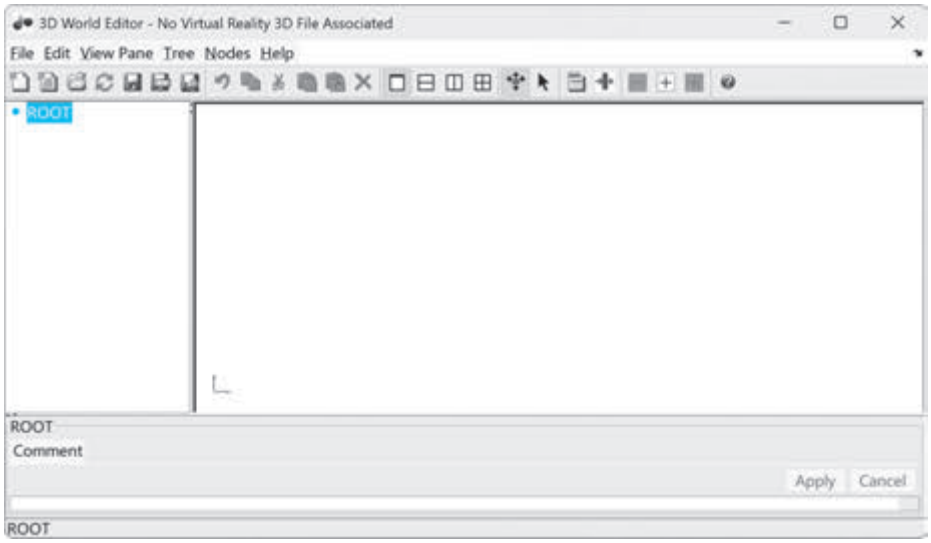
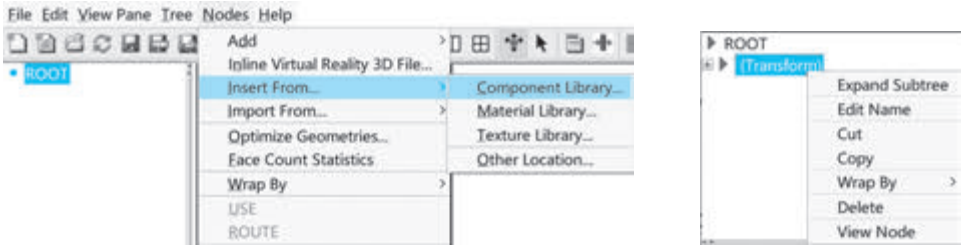


图 5-62　Simulink 三维动画场景编辑界面

Component Library(组件库)”菜单项添加预设模型。选择该菜单项将给出列表框,给出预设模型的分类,如 Aircrafts(飞机)、Animals(动物)、Architecture(建筑)、Plants(植物)等。



(a) 节点插入菜单　　(b) 节点菜单

图 5-63　相关的菜单

打开 Aircrafts,则展开下一级的模型库,包括如 Boeing_737.x3d 等具体飞机模型。选择 Boeing_737.x3d,则将在编辑界面上添加一个波音 737 模型,并在左侧 ROOTS 树状结构中会自动添加一个项目 Transform。右击 Transform,将打开如图 5-63(b)所示的菜单。

选择 Edit Name(修改名称)菜单项,可以为该模型添加或修改名称。可以为其添加名称 plane,以后可以通过名称修改飞机模型的信息。可以选择 Expand Subtrees(展开下级树)菜单项,展开下级的树状结构,图 5-64 中给出了部分树状结构和模型窗口。可以修改 center(中心),将飞机的初始位置设置为 (100,100,0),还可以选择 View Node(显示节点)菜单项,显示飞机模型。

图 5-64　树状结构与模型窗口

用类似的方法可以从 Plants 模型组中添加一棵树,命名为 tree。因为树本身是不动的,所以以后

仿真时不必修改其center、rotation等属性。

建立了三维动画场景模型后，可以将其存入文件my_world.x3d。

5.5.3 Simulink下的三维动画场景驱动

虽然静态的X3D模型可以通过其内置的脚本节点（如Script、TimeSensor）实现动画，但这要求用户深入掌握X3D/VRML语言的编程语法，过程较为复杂。对于工程仿真而言，更高效、直观的方法是使用Simulink 3D Animation模块集。我们可以将X3D模型导入Simulink环境，然后通过Simulink模块和信号流来直接驱动模型中物体的位置、旋转、颜色等属性，从而将动态仿真逻辑与三维可视化无缝集成。

Simulink的三维动画模块集提供的模块使虚拟现实和三维动画处理变得很直观。该模块集提供的模块如图5-65所示，包括VR Sink（虚拟现实显示）模块、VR To Video（虚拟现实存影像文件）模块，还提供了Joystick Input（操纵杆输入）模块和Space Mouse Input（空间鼠标输入）模块，也提供了VR Text Output（虚拟现实上叠印文字）模块和VR Tracer（虚拟现实跟踪器）模块，以后可以使用这些模块实现虚拟现实场景的显示与操作。

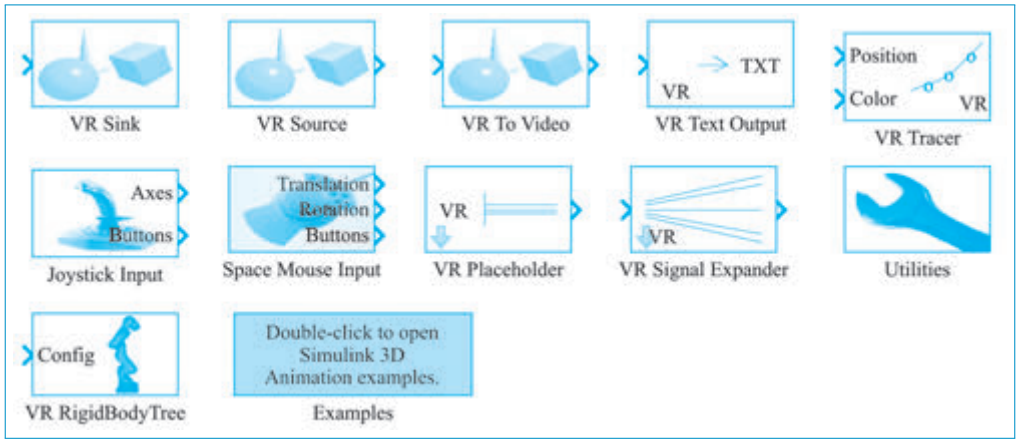


图5-65 Simulink三维动画模块集

例5-30 利用前面建立的my_world.x3d场景文件，建立Simulink模型，演示飞机绕树飞行的虚拟现实显示。

解 假定飞机从初始位置、以大树的中心为圆心、按上升的螺旋函数围绕大树飞行，则可以用下面的思路来编写程序，将仿真结果按虚拟现实的方式显示出来。

已知起始点为$(-4.7,-0.6,1)$，圆心位置为$(0,0,-9)$，并假设飞机的运动轨迹为

$$x=15\cos(t+118^\circ),\ y=-0.6+0.1t,\ z=-9+15\sin(t+118^\circ) \tag{5-5}$$

其中，参数变量$t\in(0,360^\circ)$。注意，在MATLAB下计算正弦数据需要使用弧度单位，所以应该进行变换。用下面的命令绘制出轨迹的三维图，如图5-66所示。注意，这里的坐标系是根据右手法则选定的，与常规三维坐标系是不一样的（参考坐标轴的定义）。

```
>> t=0:.1:2*pi; t0=118*pi/180; %设置t向量,将角度变为弧度
   x=15*cos(t+t0); y=-0.6+0.1*t; z=-9+15*sin(t+t0); plot3(x,z,y)
   set(gca,'XDir','reverse','YDir','reverse','box','off');
   grid, view(-67.5,52)
```

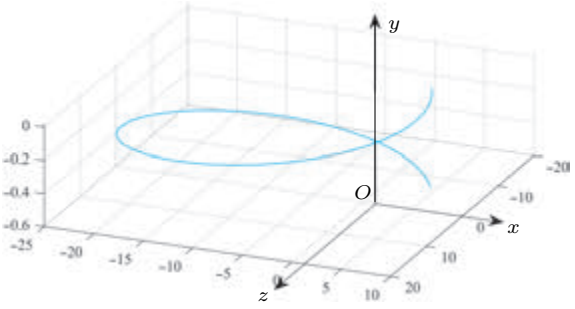


图5-66 飞行轨迹三维显示

利用Simulink环境，可以建立一个如图5-67所示的带三维动画显示的仿真框图，该框图使用了三维动画模块集的VR Sink模块来处理三维动画显示问题。

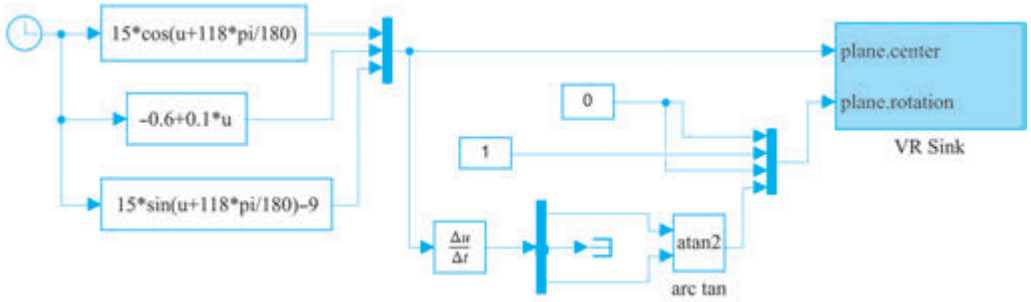


图5-67 飞机飞行仿真模块框图(文件名:c5mvrs1.slx)

将VR Sink模块复制到仿真模型时，该模块并没有输入端口，因为尚未建立该模块和场景文件之间的关联。双击该模块，将得出如图5-68所示的对话框，可以首先在左侧的Source file(源文件)编辑框中将my_world.x3d场景文件关联起来，这时，右侧的Virtual World Tree(虚拟世界树)中将出现相关属性的树状结构。因为想用仿真驱动plane对象，则应该展开该对象，选中其center和rotation属性，这样该模块就会自动生成两个输入端口plane.center和plane.rotation，用户可以最终构造如图5-67所示的仿真框图，也可以直接启动仿真过程，观察飞机飞行的动画。

具体模型的解释：模型的center属性由$[x,y,z]$向量驱动，这个向量是按式(5-5)的方式，由时钟模块驱动的。模型的rotation属性的具体描述方法是$[x_0,y_0,z_0,\theta]$，其中，$[x_0,y_0,z_0]$为转动轴的向量描述，在本例中是按y轴转动，故应该设置为$[0,1,0]$。α是依照右手法则的旋转角度(单位为弧度)，所以，可以考虑利用$\text{atan2}(\dot{x},\dot{z})$来计算旋转角度，$\dot{x}$、$\dot{z}$为当前坐标对时间的导数。

5.5.4 新版本的三维动画处理

前面介绍的基于VRML、X3D文件及vredit编辑器、VR Sink模块的工作流程，该方法适用于MATLAB R2024b及更早版本。自R2025a版本起，MathWorks已正式移除该传统工作流，转向基于sim3d框架与专业建模工具集成的现代三维仿真方案。

在MATLAB R2025a中，带有VR Sink模块的模型仍然能够使用，不过vredit编辑器不再支持，取而代之的是Epic Games公司的专业场景建模工具Unreal Engine(虚幻引擎)，新方法在渲染质量、开发效率和物理仿真方面有显著提升。

事实上，MATLAB从R2019a版开始引入sim3d框架的雏形。首次在Automated Driving Toolbox(自动驾驶工具箱)中提供与虚幻引擎的集成，用于车辆仿真，但当时的功能还比较专

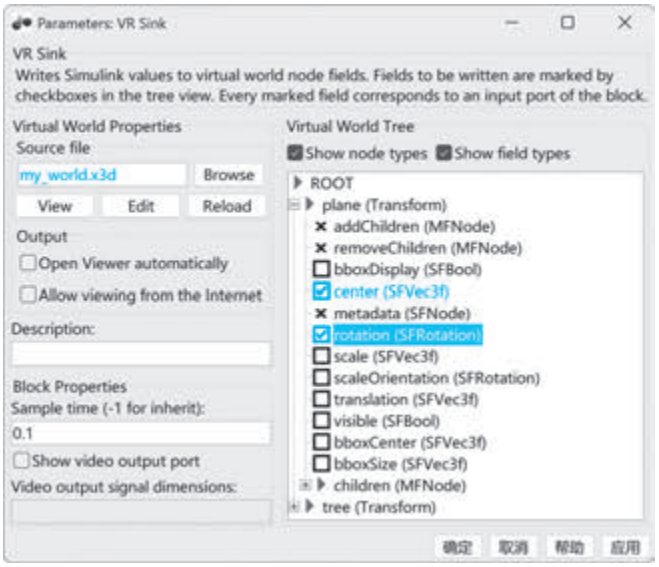


图 5-68 虚拟现实输入模块对话框

用化。从R2020b版本开始逐步推广与强化。

一般情况下，可以有两种途径实现Simulink控制的三维动画显示，一种方法是利用虚幻引擎这种专业级场景建模工具创建三维动画场景文件，然后由Simulink控制这样的文件，获得三维动画演示的效果；另一种是利用Simulink提供的三维动画建模工具生成模型，然后由Simulink驱动三维模型，用简单几何形状代表复杂模型，例如，用长方体表示飞机模型，通过仿真观察动画显示效果。从应用角度看，前一种方法适合于生成虚拟现实的逼真环境，后一种适合于测试控制算法。本节只给出后一种方法的简单介绍与演示。

在MATLAB的命令行窗口中给出`sl3dlib`命令，则打开新版的Simulink三维动画模块集，该模块集只有一个Simulink 3D模块组，双击则展开如图5-69所示的下一级模块组。

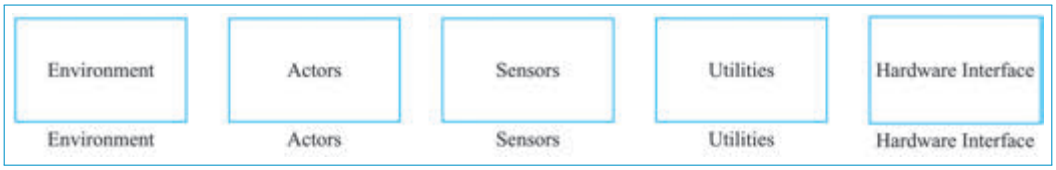


图 5-69 新版 Simulink 三维动画模块集

在该模块集中，Environment（环境）模块组只有一个Simulation 3D Scene Configuration（三维场景设置）模块，该模块是整个现代三维动画工作流的总控中心，负责配置与管理Simulink仿真所要驱动的三维虚拟场景，在三维动画的Simulink建模中，必须使用该模块。

单击Actors（角色）图标，则展开如图5-70所示的模块组。该模块组的模块如下：

- Simulink 3D Actor（通用的角色），可以用简单形状描述角色。
- Simulation 3D Vehicle with Ground Following（带地面追随的车辆），表示专用车辆角色，内置地面追随和简单车辆动力学，通过油门、转向等高层指令控制。
- Simulation 3D Bicyclist（骑行者），特指骑自行车的人，内置踩踏、转向等预设动画。
- Simulation 3D Pedestrian（行人），特指步行的人，内置行走、站立等预设动画。

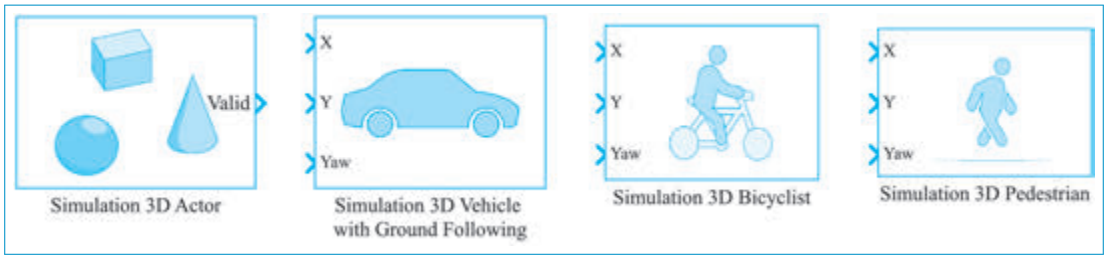


图5-70 角色模块组

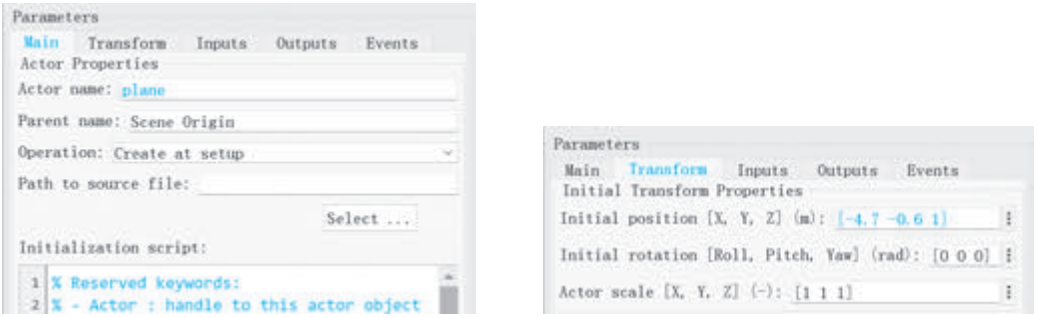
例5-31 试用新的三维动画模块集构建例5-30中的仿真模型。

解 打开空白的Simulink模型窗口，在其中粘贴Simulation 3D Scene Configuration模块，双击该模块，在Scene source(场景源)列表框中选择Default scene(预设场景)。从角色模块组复制两个Simulink 3D Actor模块，分别改名为plane和tree。下面主要考虑plane角色的参数修改。

双击该模块，则打开如图5-71(a)所示的参数对话框，这里显示的是Main选项卡，可以设置模块的名称为plane，并在Initialization scripts(初始化代码)编辑框中给出如下命令

```
createShape(Actor,'cone',[1.5,0,3]);
```

表示将飞机设置成圆锥体，其高为1.5 m，上底与下底半径分别为0 m和3 m。单击Transform(变换)选项卡，得出如图5-71(b)所示的对话框，先按图中的数值设置Initial position(初始位置)，再将Initial rotation(初始旋转姿态)设置成$(0,0,\pi/2)$，表示按z轴旋转。



(a) Main选项卡　　(b) Transform选项卡

图5-71 飞机角色的参数对话框

除了这些参数外，还需要为飞机模块添加输入端口，以便于用Simulink信号驱动三维动画。这需要首先从5-71(a)的对话框进入Inputs(输入)选项卡，如图5-72所示。选中plane.Translation和plane.Rotation项目，移动到Input ports(输出端口)，这样模块就增加了输入端口。

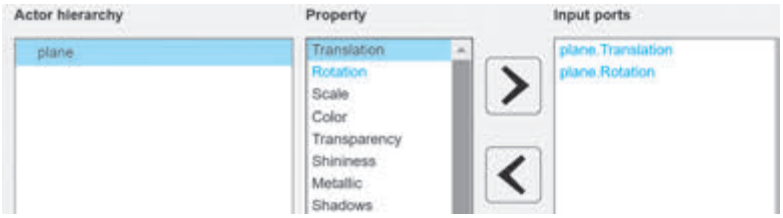


图5-72 飞机角色的输入端口添加

参考图5-67中给出的早期版本模型，可以得出新版本可用的仿真模型，如图5-73所示。plane模块的输入格式比较严格，需要1×3行向量，而不是Mux模块输出的列向量，所以需要用Transpose(转

置）模块进行变换。旋转属性的定义与早期版本也是不同的，以前需要$[0,1,0,\alpha]$这样的四元数组，而新版本需要的是1×3行向量定义的Euler角，所以新模型中做了相应的修改。此外，新版本的坐标轴顺序定义也和早期版本不同，所以在模型中也给出了相应的变化。

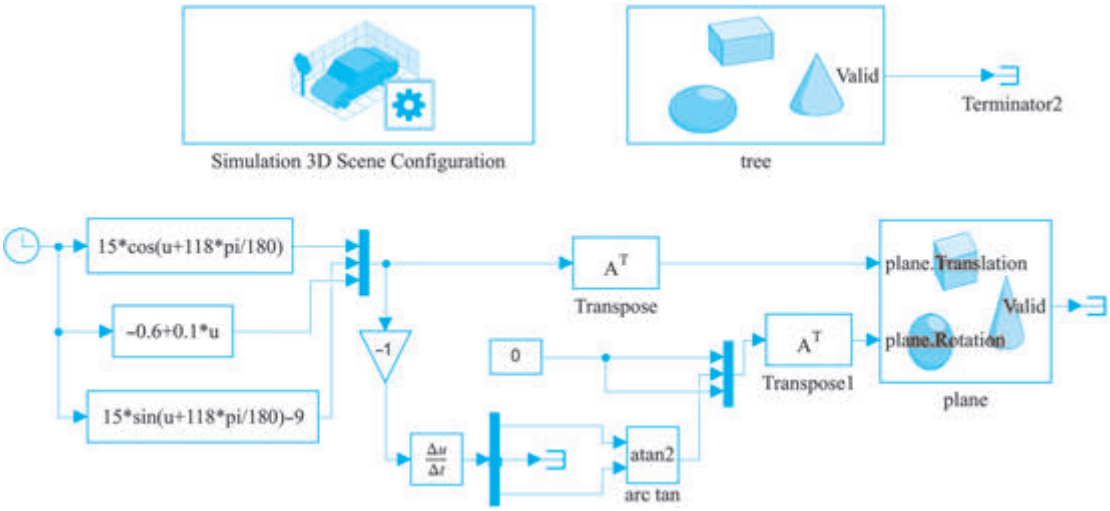


图5-73 飞机飞行仿真模块框图（文件名：c5mvrs1_new.slx）

5.6 子系统与模块封装技术

在系统建模与仿真中，经常遇到很复杂的系统结构，难以用一个单一的模型框图进行描述。通常地，需要将这样的框图分解成若干具有独立功能的子系统，在Simulink下支持这样的子系统结构。另外，用户还可以将一些常用的子系统封装成为一些模块，这些模块的用法也类似于标准的Simulink模块。更进一步地，还可以将自己开发的一系列模块做成自己的模块组或模块集。本节将系统地介绍子系统的构造及应用、模块封装技术和模块库的设计方法，并通过一个较复杂系统的例子来演示子系统的构造和整个系统的建模。

5.6.1 子系统的处理

要建立子系统，首先需要给子系统设置输入和输出端。子系统的输入端由Sources模块组中的In来表示，而输出端用Sinks模块组的Out来表示。在输入端和输出端之间，用户可以任意地设计模块的内部结构。

当然，如果已经建立起一个方框图，则可以将想建立子系统的部分选中，具体的方法是：用鼠标左键单击要选中区域的左下角，拖动鼠标在想选中区域的右上角处释放，则可以选中该区域内所有的模块及其连接关系。用鼠标选择了预期的子系统构成模块与结构之后，可以通过选择“基于所选内容创建子系统”命令来建立子系统。如果没有指定输入和输出端口，则Simulink会自动将流入选择区域的信号依次设置为输入信号，将流出的信号设置成输出信号，从而自动建立起输入与输出端口。

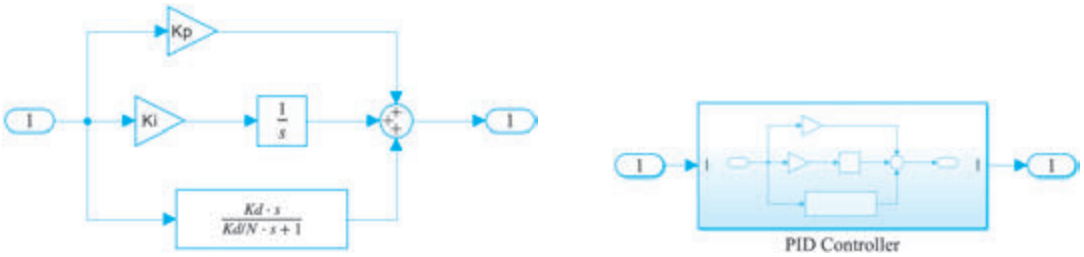
例5-32 PID控制器是在自动控制中经常使用的模块，在工程应用中其标准的数学模型为：

$$U(s)=\left(K_{\mathrm{p}}+\frac{K_{\mathrm{i}}}{s}+\frac{sK_{\mathrm{d}}}{1+sK_{\mathrm{d}}/N}\right)E(s) \tag{5-6}$$

其中采用了一阶环节来近似纯微分动作，为保证有良好的微分近似的效果，一般选 $N \geqslant 10$。试搭建一个PID控制器子系统模块。

解 Simulink的Continuous模块组提供了强大的、可以直接使用的PID控制器模块，所以，这里只是演示一个简单的PID控制器建模与封装方法。可以由Simulink环境直接建立PID控制器的模型，如图5-74(a)所示。注意，这里的模型含有4个变量，即Kp、Ki、Kd和N，这些变量应该在MATLAB工作空间中赋值。

绘制了原系统的框图，可以选中其中所有的模块，例如，可以按Ctrl+A组合键选择所有模块，也可以用鼠标拖动的方法选中。这样可以用“基于所选内容创建子系统”快捷菜单构造子系统，得出的子系统框图如图5-74(b)所示，在子系统图标中显示了其内部结构。双击子系统图标可以打开原来的子系统内部结构窗口，如图5-74(a)所示。



(a) PID控制器模型(文件名:c5mpid1.slx)　　(b) 生成的子系统示意图(c5mpid1a.slx)

图5-74　PID控制器的Simulink描述

从前面的介绍看，因为“基于所选内容创建子系统”快捷菜单允许用户从选择的模块提取子系统模型，所以，可以利用该功能直接从现有Simulink模型中提取整个模型的一部分，并创建子系统模块。

例5-33　在工业控制中，一般的PID控制器带有饱和非线性环节，其控制结构如图5-75所示。假设受控对象模型由例4-9给出，且PID控制器参数为 $K_p = 0.998$，$K_i = 0.333$，$K_d = 0.0809$，饱和非线性的幅值为 $|u(t)| \leqslant 0.5$。试搭建Simulink仿真模型，并构造子系统形式。

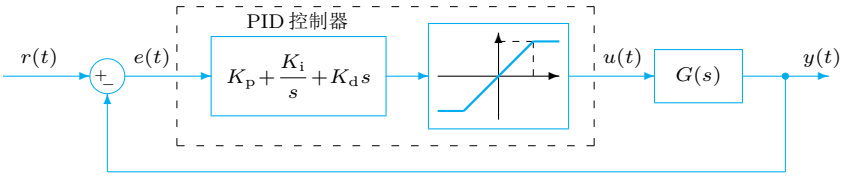


图5-75　PID控制系统的框图表示

解 可以直接构造如图5-76所示的Simulink仿真模型。

可以用鼠标画线的方法同时选中PID控制器模块与饱和非线性模块，如图5-77(a)所示。可以看出，被选中的模块与区域用不同的线宽标出，以示与其他未选中模块的区别。

选中了相关的模块与结构之后，再选择“基于所选内容创建子系统”快捷菜单项，则可以构造如图5-77(b)所示的仿真模型。可以看出，选中的诸多模块就变成了单个子系统模块，其内部结构仍然由缩略图表示。

构造了子系统模型后，选中子系统模块，再选择快捷菜单中的“子系统和模型引用 ▸ 展开子系统”菜单项，则可以把选中的子系统模块直接展开，还原图5-77(a)中给出的模型。

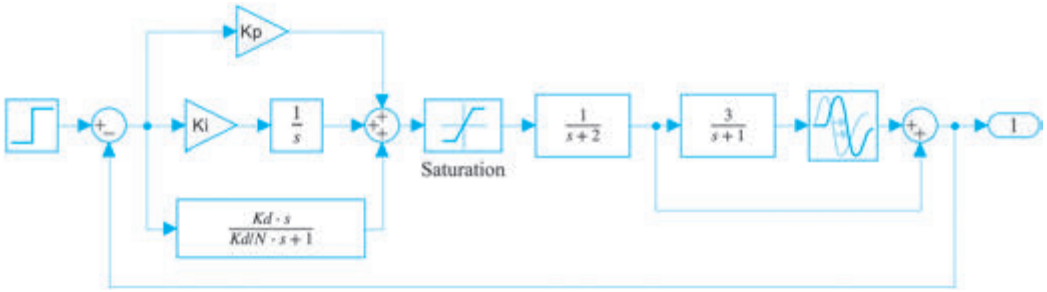
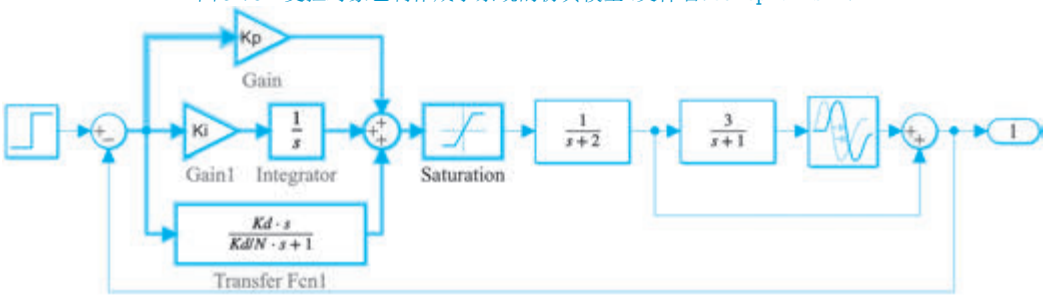
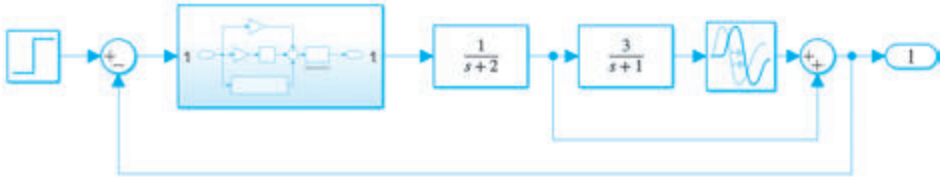


图5-76 受控对象也制作成子系统的仿真模型(文件名:c5mcpid1.slx)



(a) 选中PID控制器的模型



(b) 提取的PID控制器子系统(文件名:c5mcpid2.slx)

图5-77 由仿真模型中提取并制作子系统

用类似的方法还可以将受控对象模型制作成子系统,这时的仿真模型如图5-78所示。可以看出,利用子系统的方法,可以使复杂系统的模型结构显得更简洁。如果需要修改或观察某个子系统,双击该图标就可以打开子系统模型,这也使得Simulink建模与维护变得更容易。

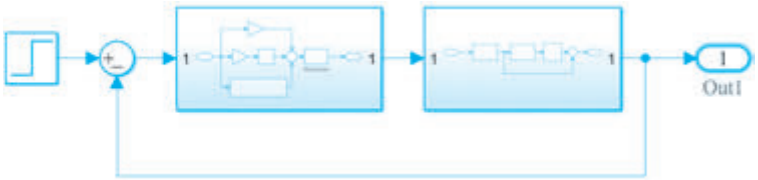


图5-78 受控对象也制作成子系统的仿真模型(文件名:c5mcpid3.slx)

5.6.2 条件执行子系统

在Simulink中,允许某个子系统在给定的控制信号下执行,这样的子系统称为条件执行子系统(conditionally executed subsystems),当前版本共支持下面3种控制结构。

- **使能子系统**(enabled subsystem)。将子模块条件信号称为控制信号,控制信号分成允许(enable)和禁止(disabled)两种。在允许信号控制下(Simulink的约定为,当控制信号为正时将模块设置为允许状态,否则为禁止状态),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

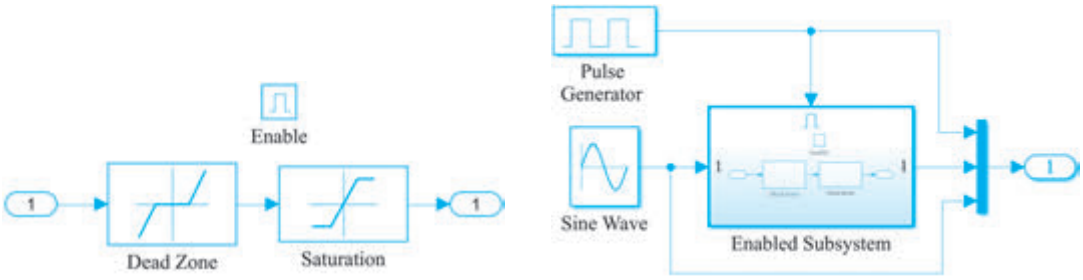
继续保持禁止前的信号或复位子系统,强制使其输出零信号。

- **触发子系统**(triggered subsystem)。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,然后保持子系统输出的状态,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。
- **使能触发子系统**(enabled and triggered subsystem)。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。

下面通过例子来演示条件执行子系统的构造和功能,并介绍有关触发的概念。

例5-34 考虑由死区和饱和非线性环节串联的结构,试构造一个使能子系统。

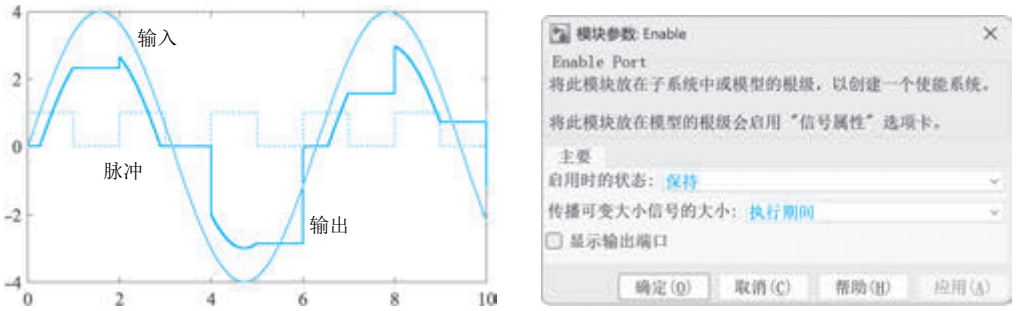
解 可以按如下的方式建立起使能子系统:首先打开一个空白的模型窗口,将Ports & Subsystems模块组中的Enabled Subsystem模块拖动到模块组中。双击该模块,则将打开子系统编辑窗口,在该窗口中将所需的非线性模块建立起来,并设置死区模块的参数为$(-1,1)$,饱和模块的参数为$(-3,3)$,则可以建立起如图5-79(a)所示的子系统。



(a) 使能子系统(c5mcon1.slx) (b) 仿真框图(c5mcon1a.slx)

图5-79 使能子系统框图

在模型窗口中给子系统施加一个幅值为4的正弦输入信号,并给使能控制信号加一个脉冲信号(周期设置为2s,脉冲宽度设置为50,表示50%),这样可以得出如图5-79(b)所示的仿真框图。选择定步长仿真,并将仿真步长设置为0.02s,则得出如图5-80(a)所示的仿真结果。为了更好地观测输出信号,加粗了该曲线,从几条曲线的比较中可以看出使能信号的作用。



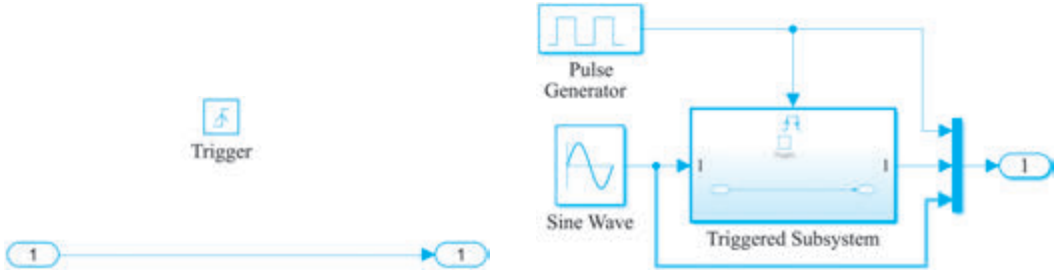
(a) 仿真结果 (b) 使能信号对话框

图5-80 使能信号仿真结果与设置

进入子系统模型,双击使能图标,则将打开如图5-80(b)所示的对话框,在其中可以选择使能"启用时的状态",其值可以选择"复位"和"保持",不过在这个静态非线性系统中两者没有区别。

例5-35 观察触发子系统的作用与效果。

解 重新改写子系统的内部结构，让其输入端直接连接到其输出端，这样就可以将Triggered Subsystem(触发子系统)模块复制到模型编辑窗口。双击该子系统图标，则将打开如图5-81(a)所示的子系统模型，这样就可以在模型编辑窗口中构造主系统模型，如图5-81(b)所示。

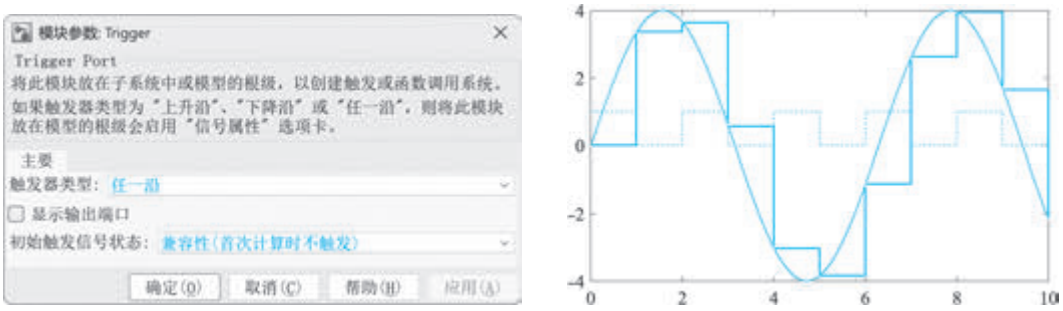


(a) 触发子系统(c5mcon2.slx) (b) 仿真模型(c5mcon2a.slx)

图5-81 触发子系统的模型

双击子系统中的触发器图标，则将打开如图5-82(a)所示的对话框，可以看出，在“触发器类型”下拉列表框中，可以选择“上升沿”、“下降沿”、“任一沿”及“函数调用”等。

启动仿真过程，再给出 `stairs(tout,yout)` 命令，则得出如图5-82(b)所示的曲线图。从效果上看，这相当于在输入信号后面接一个零阶保持器模块。



(a) 触发器对话框 (b) 上升沿触发仿真结果

图5-82 触发器设置与仿真

如果将触发器改成“下降沿”，则可以得出相应的结果，读者可以自行观察。

Simulink还允许使能端口和触发端口共同控制子系统，在系统处于使能状态时，触发事件发生则将激活子系统；如果系统处于非使能状态，则将忽略触发信号的作用。

Simulink中还提供了更复杂的流程控制，如转移、开关和循环等控制结构，这些结构是依赖Stateflow构造的，这些模块还可以用语句实现。

5.6.3 模块封装技术

前面介绍的子系统是把共性的部分独立出来，做出可独立模块，简化整个仿真系统的结构。不过，如果仿真模型中有两个参数不同的PID控制器，则不能采用PID控制器的子系统模块，因为这两个模块参数将保持一致。所以，需要对PID控制器子系统进行封装，制作成可重用的独立模块。

所谓封装(masking)模块，就是将其对应的子系统内部结构隐藏起来，以便访问该模块时只出现参数设置对话框，模块中所需要的参数可以由这个对话框来输入。其实Simulink中大多

数的模块都是由更底层的模块封装起来的，如传递函数模块，其内部结构是不可见的，它只允许双击打开一个参数输入对话框来读入传递函数的分子和分母参数。在前面介绍的PID控制器中，也可以将它封装起来，只留下对话框输入该模块的4个参数。

如果想封装一个用户自建模型，首先应该用建立子系统的方式将其转换为子系统模块，选中该子系统模块的图标，再选择“封装 ▸ 创建封装”快捷菜单项，则可以打开如图5-83所示的模块封装编辑器。

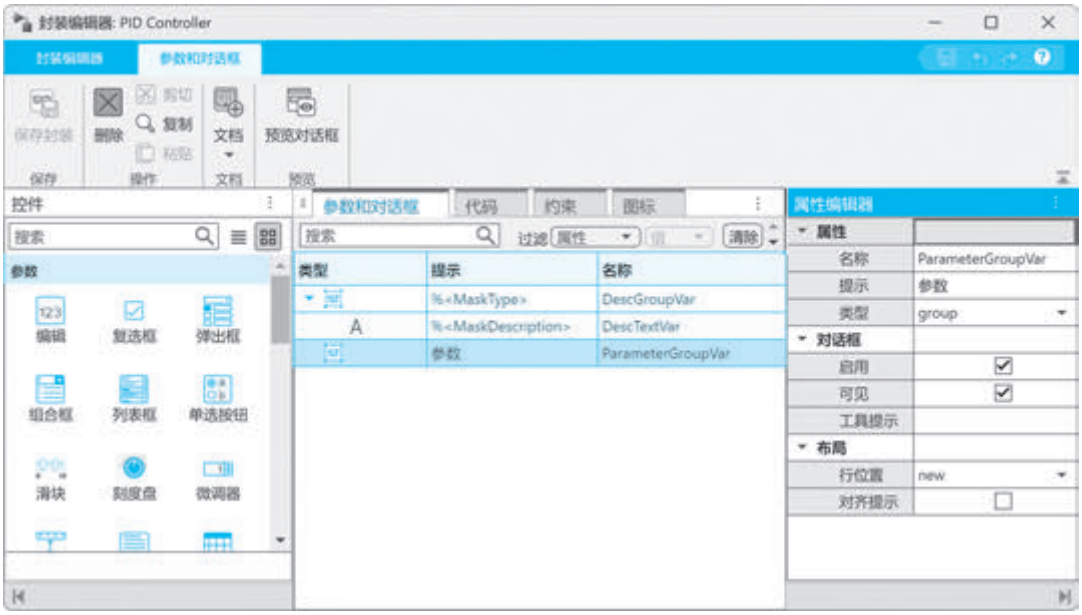


图5-83 Simulink的封装编辑器

该界面中部区域有四个标签，“参数和对话框”、“代码”、“约束”和“图标”，每个标签对应一个选项卡。当前的选项卡用于封装模块的参数对话框设计。在对话框左侧区域提供了各种控件，如“编辑”“复选框”“弹出框”等，可以用于参数对话框的设计。

下面将通过例子介绍模块的封装方法。

例5-36 考虑例5-32中设计的PID控制器子系统模块。试封装并构造可重用PID控制器模块。

解 在研究封装方法之前应该先了解参数对话框的需求。由式(5-6)可见，PID控制器有四个参数，K_p、K_i、K_d和N，前三个参数可以由编辑控件描述，后一个可以由列表框描述。

拖动左侧的“编辑”控件到中区的“参数”栏目下，就可以在参数对话框再添加一个编辑框。用类似的方法添加三个编辑框和一个列表框，如图5-84所示。选择第一个参数#1，可以在右侧的“名称”编辑框中填写变量名，在“提示”编辑框填写提示用文字信息，再在右侧属性编辑器的“值”编辑框填写参数值，这样就可以完成第一个参数的设计。用类似的方法还可以设置其他三个参数，如图5-85所示。在设计下拉式列表框时，右侧“类型”条目的下面多了一个“类型选项”条目，在其中输入所需的列表选项，如本例填写10、100和1000。

用户还可以在模块打开之前预先执行某些命令，这些命令可以由“代码”选项卡设置。单击“代码”选项卡，则对话框的编辑区域如图5-86所示。这里给出的是空白的类设置框架，用户还可以在这个区域直接给出想运行的代码。在这个编辑框中，允许使用左侧列出的变量。

图5-84 参数对话框设计雏形

图5-85 参数对话框的设计

图5-86 初始化设置对话框

设置完封装模块后，双击该模块图标，则打开如图5-87所示的模块参数对话框。

图5-87 封装模块的参数对话框（文件名：c5mpid2.slx）

5.6.4 封装模块的图标设计

用户还可以利用封装编辑器为封装模块设计图标。单击封装编辑框的“图标”选项卡，则图标设计区域如图5-88所示。其中，左侧区域提供了一系列图标设计支持的MATLAB语句，

可以将所需的语句在中间的栏目写出即可。

图5-88 图标设计区域

一般情况下,可以使用下面三种方法绘制图标。

(1) **图形绘制**。允许在该模块图标上绘制曲线或颜色填充片。由MATLAB的`plot()`函数画出线状的图形。注意,这里的`plot()`命令不是真正的MATLAB命令,它的调用格式为`plot(`x_1`,`y_1`,`x_2`,`y_2`,···)`,不像MATLAB的`plot()`函数那样,支持曲线绘制选项。

此外,还可以由`patch()`函数绘制颜色填充片,其调用格式为

`patch([`$x_1,x_2,\cdots,x_n,x_1$`],[`$y_1,y_2,\cdots,y_n,y_1$`],`颜色`)`

其中,这里给出的两个向量描述的是封闭曲线的坐标,其内部用指定颜色填充。"颜色"可以为`'red'`(红色)这样的保留颜色,也可以为$[r,g,b]$这样的三原色颜色分量。这三个分量的取值范围均为$[0,1]$。

在绘制图形或文字时,还允许修改绘制的颜色。例如,若使用`color(`颜色`)`命令,则后续画图与文字都将按照设定的颜色给出。

(2) **文字描述**。使用`disp()`语句可以在图标上叠印文字,该命令允许文字的多行显示,用户可以在要显示的文字字符串中用`\n`表示换行。比较新的版本还允许用户使用`text()`命令,在指定位置显示文字,其具体调用格式为`text(`x,y`,`字符串`)`,其中,(x,y)为显示字符串的坐标,其尺度与前面`plot()`语句是一致的。

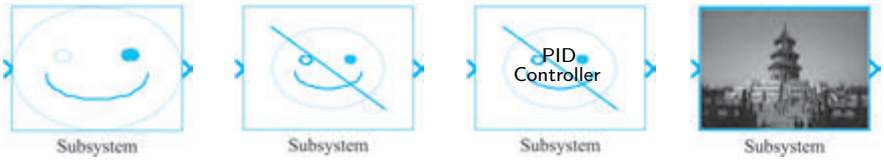
(3) **图像文件**。MATLAB允许用户使用`imread(`文件名`)`语句将一个已知的图像文件读入MATLAB工作空间中,这时,可以调用`image()`函数就能显示图片。可以用这样的方式在图标上显示图片。

如果想在设计图标时使用"代码"编辑区中的变量,则应该选择图5-88对话框右下角的"运行初始化"列表框,将其从Off改成On,否则,绘制图标时将不能识别初始化变量。

例5-37 试为封装的PID控制器模块设计图标。

解 如果想在图标上画出一个"笑脸",则可以采用下面的MATLAB命令,分别绘制出四条曲线,其中外部画一个单位圆表示"脸";绘制两个小圆,半径为0.1,圆心分别在$(-0.4,0.2)$和$(0.4,0.2)$处,表示眼睛;在底部画一个半椭圆,表示嘴。特别地,右侧的眼睛用`patch()`命令绘制填充圆,颜色为红色。这时设计的图标如图5-89(a)所示。图标设计完成后,其缩略图在图5-88界面的"预览"区域自动显示出来(这里只显示局部,用下拉滚动杆可以预览剩余的部分)。

```
t=linspace(0,2*pi,30); t1=linspace(0,pi,30);
plot(cos(t),sin(t),-0.4+0.1*cos(t),0.2+0.1*sin(t))
patch(0.4+0.1*cos(t),0.2+0.1*sin(t),[1,0,0]) %红色填充图
plot(0.6*cos(t1),-0.1-0.4*sin(t1))            %默认黑色曲线
```



（a）笑脸图标　（b）再添加一条反斜线　（c）叠印文字　（d）绘制图像

图5-89　笑脸图标的实现（文件名：c5mpid3.slx）

如果想在笑脸上加一条斜线，则可以直接加一行命令

```
plot([-1,1],[1,-1]), plot(-1.5,-1.5), plot(1.5,1.5)
```

这时，会自动在原来的笑脸上叠印一条斜线。另外，由于后两条语句的加入，坐标系范围被设置成$-1.5 \leqslant x,y \leqslant 1.5$。这样设置的图标如图5-89(b)所示。注意，在绘制图标时不能使用`hold on`命令，也不能使用`line()`语句（该语句在此情景不绘图）。`plot()`命令和MATLAB的`plot()`函数不一样，绘图时不清屏，会直接将新的曲线叠印在现有的曲线上。

如果在前面的绘图命令后加一条指令`disp('PID\nController')`，则可以在图标上叠印文字，如图5-89(c)所示。若给出命令`image(imread('tiantan.jpg'))`，则会读入`tiantan.jpg`文件，并在图标上显示该图像，如图5-89(d)所示。可以看出，绘制图像后将覆盖全部的绘图命令。

例5-38　5.2节介绍的分段线性的静态非线性模块的建模。试封装并构造可重用模块。

解　单值非线性可以由查表模块直接实现，而双值非线性环节需要拆分成两个查表模块，再配以开关模块实现。现在考虑建立一个统一的分段线性非线性模块。

可以提出下面的协议。非线性模块带有两个参数$\boldsymbol{x}$和$\boldsymbol{y}$。如果这两个参数为行向量，则非线性环节为单值非线性；如果这两个参数为两行的矩阵，则对应的非线性环节为双值非线性环节。另存`c5mloop2.slx`文件为`c5mloop2a.slx`，将“u上升”模块的参数修改为变量`x1`和`y1`，“u下降”模块的参数修改为`x2`和`y2`，并将正弦输入模块替换成输入端口，在此基础上建立统一的非线性封装模块。

先选中所有模块构造子系统，再由生成的子系统模块开启封装编辑器。在编辑器中，由“参数和对话框”选项卡定义两个输入变量，`xData`与`yData`，如图5-90所示。

图5-90　封装编辑器界面

这样就有一个很明显的问题：如何把对话框输入的`xData`与`yData`数据分派到模型的$\boldsymbol{x}_1$、$\boldsymbol{y}_1$等向量？要解决这样的问题，需要编写封装模块的初始化程序。可以将下面的命令填写到“代码”选项卡。这样，就解决了模型变量的分派问题。每次参数对话框数据有变化，则会自动完成变量的分派。

```
if size(yData,1)==1 %如果输入参数是行向量,则为单值函数
   x1=xData; x2=x1; y1=yData; y2=y1;
else                %否则为双值函数
   x1=xData(1,:); x2=xData(2,:); y1=yData(1,:); y2=yData(2,:);
end
```

第二个问题:如何绘制模块的图标?有了x_1、y_1等向量,在“图标”选项卡下似乎可以用plot(x_1,y_1,x_2,y_2)绘制双值非线性图标。不过直接这样做会出现错误,提示x_1等变量未定义。观察“图标”选项卡的设置可见,左下角附近的“运行初始化”列表框,默认的选项是Off,因此初始化信息没有被正确地传递到这个选项卡,应该将其设置为On。这样,由plot()函数就可以正确绘制图标了。为更好绘制曲线图标,应该在“图标”编辑框中填写如下的命令,增加绘图裕量:

```
plot(x1,y1,x2,y2)
dX=max(xData(:))-min(xData(:)); dY=max(yData(:))-min(yData(:));
plot(min(xData(:))-0.08*dX,min(yData(:))-0.08*dY)
plot(max(xData(:))+0.08*dX,max(yData(:))+0.08*dY)
```

这样建立的模块既可以处理单值非线性特性,也可以处理双值非线性特性。可以看出,利用模块封装的强大功能,就可以容易地构造出可重用的Simulink模块。

封装的模块可以用“封装 ▸ 创建封装”快捷菜单项重新编辑,这将打开如图5-83所示的对话框,重复前述的步骤即可修改封装的参数。要修改模块内部的结构,则应该右击该模块,在打开的快捷菜单中选择“封装 ▸ 查看封装内容”菜单项,将打开构成该模块的子系统结构图,可以修改其实际框图。

5.6.5 组建自己的模块库

如果用户自己建立了很多封装的模块,则往往需要再建立一个模块库来存储这些模块。另外,用户也可以将常用的一组模块建立一个单独的模块库,以便自己调用。

创建模块库的方法是:选中Simulink起始页中的“空白库”图标,这样将打开一个空白的模块库窗口。可以将该模块库存为新的文件,如my_blks.slx。这样建立起来的模块库处于锁定状态,模块库中各个模块是不能进行修改或移动位置的。要想修改其中的模块,则需要使用“解锁的库”将其解锁,修改完成后存盘。

有了这样的模块库,则可以将常用的模块复制到该模块库中,能构造出如图5-91所示的自己的模块库,所以在绘制简单的框图时不必再打开Simulink,而只需在MATLAB命令行窗口中输入my_blks或open_system('my_blks')命令即可。

如果想让该模块组加入Simulink的模块库,则需要建立slblocks.m的文件。该文件有固定格式,用户可以搜索一个成型的slblocks.m文件,并将其复制到my_blks.slx所在目录,然后修改下面几条命令:

```
blkStruct.Name=sprintf('%s\n%s','Commonly Used Blocks','for Simulink');
blkStruct.OpenFcn = 'my_blks';
```

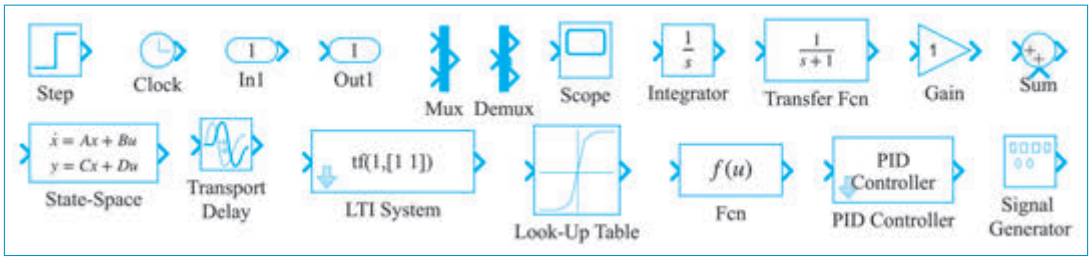


图5-91 常用模块库窗口（文件名:my_blks.slx）

5.6.6 子系统应用举例——F-14战斗机仿真

在介绍子系统和封装技术之前先看一个复杂系统的例子，该例子是在控制系统计算机辅助设计软件不是很发达的时候，由美国学者Dean Frederick等人提出的，当时是用于测试计算机辅助设计软件功能和建模准确性的，即著名的F-14战斗机模型与仿真的基准测试问题（benchmark problem）[10]。该问题提出以来，国际上很多学者用不同的算法和计算机软件陆续给出了求解方法，但有些求解方法现在看来是很烦琐的，因为已经有了新一代的计算机软件，如Simulink这样的交互绘图式的软件来表示复杂的系统模型。

例5-39 F-14战斗机基准问题的框图如图5-92所示，该系统共有两路输入信号，其向量表示为 $\boldsymbol{u}=[n(t),\alpha_{\mathrm{c}}(t)]^{\mathrm{T}}$，其中 $n(t)$ 为单位方差的白噪声信号，而 $\alpha_{\mathrm{c}}(t)=K\beta(\mathrm{e}^{-\gamma t}-\mathrm{e}^{-\beta t})/(\beta-\gamma)$ 为攻击角度命令输入信号，这里 $K=\alpha_{\mathrm{c}_{\max}}\mathrm{e}^{\gamma t_{\mathrm{m}}}$，且 $\alpha_{\mathrm{c}_{\max}}=0.0349$，$t_{\mathrm{m}}=0.025$，$\beta=426.4352$，$\gamma=0.01$。已知系统中各个模块的参数为：

$\tau_{\mathrm{a}}=0.05$，$\sigma_{\mathrm{wG}}=3.0$，$a=2.5348$，$b=64.13$

$V_{\tau_0}=690.4$，$\sigma_{\alpha}=5.236\times10^{-3}$，$Z_{\mathrm{b}}=-63.9979$，$M_{\mathrm{b}}=-6.8847$

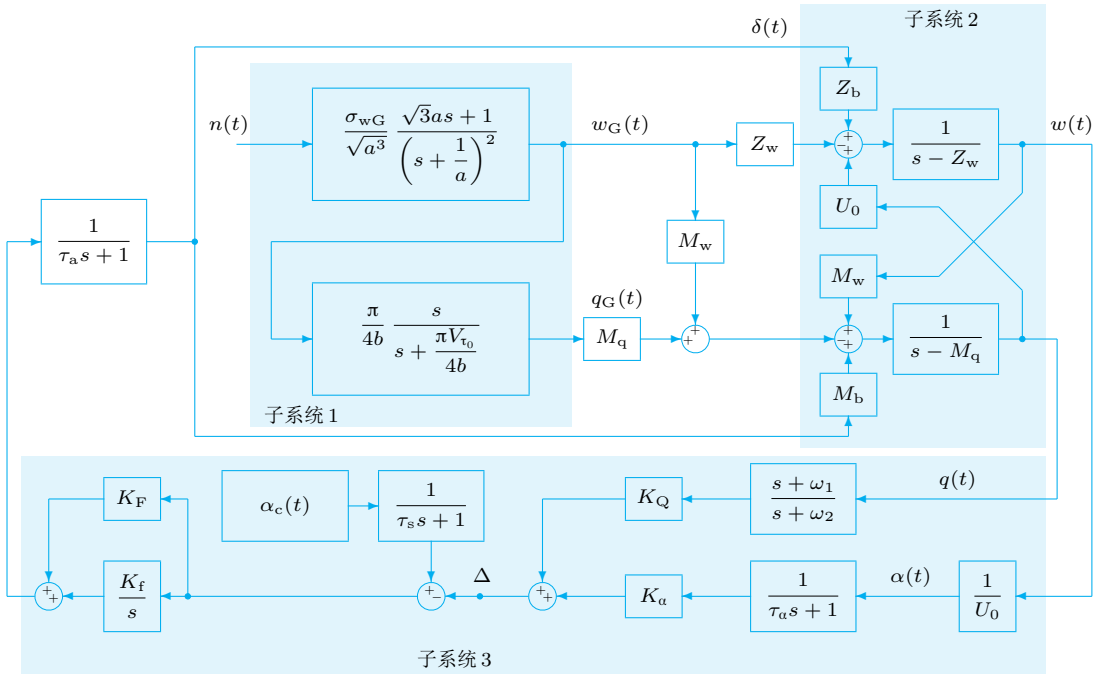


图5-92 F-14战斗机模型框图

$U_0=689.4$, $Z_w=-0.6385$, $M_q=-0.6571$, $M_w=-5.92\times10^{-3}$

$\omega_1=2.971$, $\omega_2=4.144$, $\tau_s=0.10$, $\tau_\alpha=0.3959$

$K_Q=0.8156$, $K_\alpha=0.6770$, $K_f=-3.864$, $K_F=-1.745$

可以用下面语句定义一系列MATLAB变量,其顺序与前面的变量列表完全对应:

```
tA=0.05; Swg=3.0; a=2.5348; b=64.1300;
Vto=690.4; Sa=0.005236; Zb=-63.9979; Mb=-6.8847;
U0=689.4; Zw=-0.6385; Mq=-0.6571; Mw=-0.00592;
w1=2.971; w2=4.144; ts=0.1; ta=0.3959;
KQ=0.8156; Ka=0.677; Kf=-3.864; KF=-1.7450;
g=0.01; be=426.4352; tm=0.025; K=0.0349*exp(g*tm);
```

其中,变量g和be分别对应前面的γ和β。可将这些变量写入模型的PreloadFcn属性,这样在进行仿真之前就能进行变量赋值。

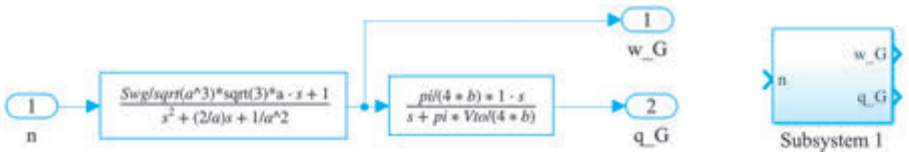
观察原系统可见,整个系统的输出有3路信号,$\boldsymbol{y}(t)=\left[N_{Z_p}(t),\alpha(t),q(t)\right]^T$,这里$N_{Z_p}(t)$信号定义为:

$$N_{Z_p}(t)=\frac{1}{32.2}\left[-\dot{w}(t)+U_0q(t)+22.8\dot{q}(t)\right] \tag{5-7}$$

试用Simulink建立整个F-14战斗机系统的仿真模型。

解 可见原系统结构较复杂,若在一个Simulink模型窗口中绘制整个Simulink仿真模型会很麻烦,得出的系统会比较凌乱,也不易维护,因此,可以采用子系统的形式,化整为零,建立整个仿真模型。可以将其分成4个子系统,其中前3个子系统按照图5-92中阴影框所示进行建模,第四个子系统描述式(5-7)中给出的信号进行建模。

首先考虑子系统1,可见该模块有1路输入信号$n(t)$,有两路输出信号$w_G(t)$和$q_G(t)$,所以可以按如图5-93(a)建立其子系统模型。选中该窗口中的所有模块,再选择“基于所选内容创建子系统”快捷菜单项,则可以得出如图5-93(b)所示的子系统图标。



(a) 子系统1　　(b) 子系统模块

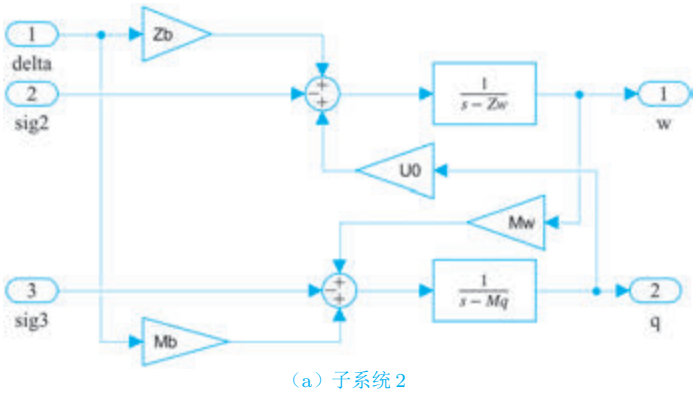
图5-93　子系统1的Simulink表示(总模型文件:c5f14.slx)

再考虑子系统2,该系统有3路输入信号,其中第一路为$\delta(t)$,第二路信号$w_G(t)$为经Z_w模块后的信号,第三路为$-(M_ww_G(t)+M_qq_G(t))$,子模块有两路输出,即$w(t)$和$q(t)$。这样就可以建立起子系统2的模型,如图5-94(a)所示,制成子系统后如图5-94(b)所示。

子系统3有两路输入信号,即$w(t)$和$q(t)$,及一路输出信号,故可以建立起如图5-95(a)所示的子系统模型,制成子系统后图标如图5-95(b)所示。

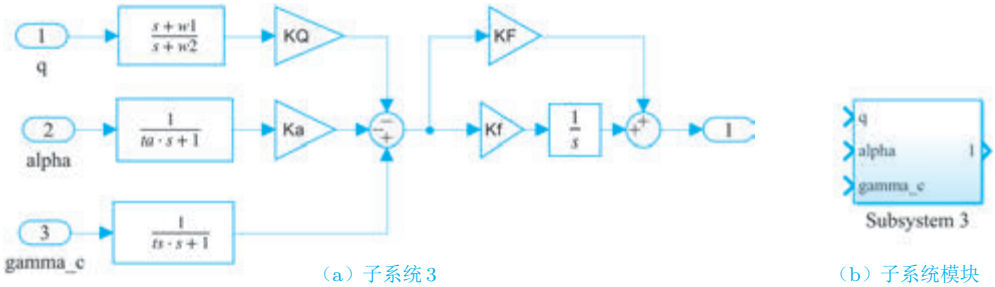
现在来建立子系统4,即用子系统的方式来表示式(5-7)。可以由图5-96(a)中所示的方式来表示该子系统。子系统存成如图5-96(b)所示的图标的形式。

建立各子系统模型后,可以比较容易地建立如图5-97所示的整个F-14战斗机系统的Simulink仿真模型。可见,采用子系统的形式可以使得原来很复杂的问题分解成各个相对简单的小块,然后将这些小块再连接成整个系统,这使得整个系统的建立和维护都变得更加简单。



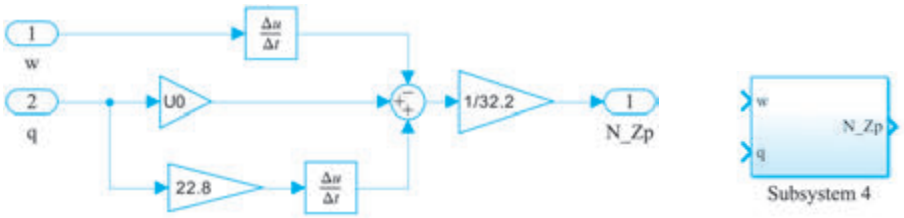
（a）子系统2　　（b）子系统模块

图5-94　子系统2的Simulink表示



（a）子系统3　　（b）子系统模块

图5-95　子系统3的Simulink表示



（a）子系统4　　（b）子系统模块

图5-96　子系统4的Simulink表示

对系统进行仿真分析，则可以得出各路输出信号，其第二列为实际的攻击角$\alpha(t)$，还可以由下面语句计算给定攻击角$\alpha_c(t)$，这两个信号的曲线如图5-98所示。

```
>> ac=K*be*(exp(-g*tout)-exp(-be*tout))/(be-g); %给定攻击角
   plot(tout,yout(:,2), tout,ac,'--')          %攻击角曲线
```

5.7 习题

5.1 用向量输入积分器的方法表示Lorenz方程的Simulink模型，并进行仿真运算，比较和原来Simulink模型的差异。

5.2 考虑Lorenz方程模型，该模型没有输入信号，假设选择其3个状态变量$\boldsymbol{x}_i(t)$为其输出信号，以β、σ、ρ和$\boldsymbol{x}_i(0)$向量为附加参数，试将该模块封装起来，并绘制在不同参数下的Lorenz方程解的三维曲线。

5.3 考虑图5-99中给出的两个Simulink模型，试分析这两个模型是否含有代数环，试进行仿真验

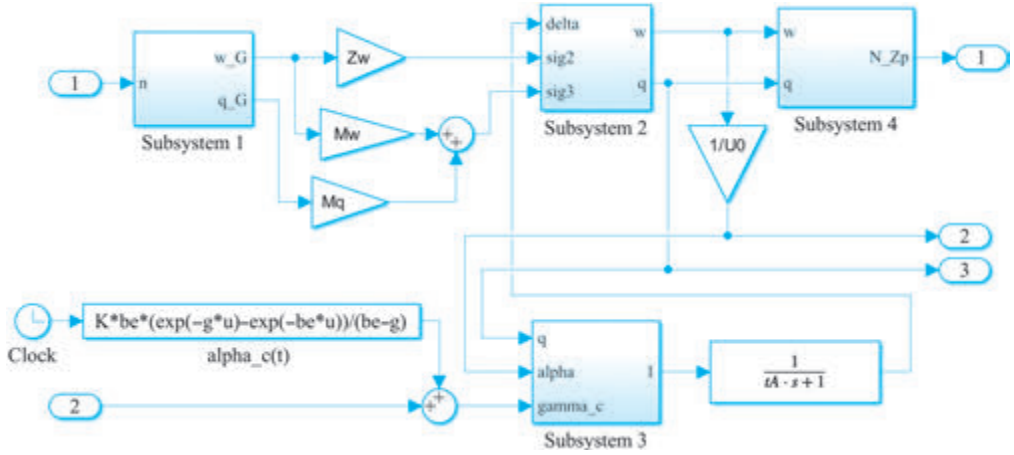


图5-97 F-14战斗机系统的Simulink模型(文件名:c5f14.slx)

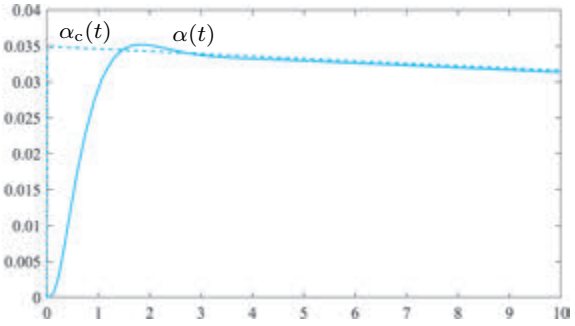
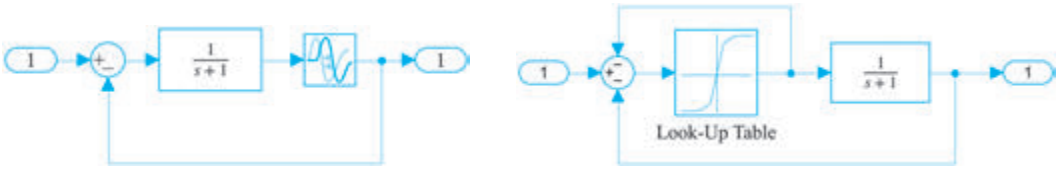


图5-98 F-14战斗机的攻击角曲线

证,并分析如果存在代数环是否影响仿真结果。



(a)系统模型1(文件名:c5exa1a.slx) (b)系统模型2(文件名:c5exa1b.slx)

图5-99 习题5.3图

5.4 考虑简单的线性微分方程 $y^{(4)}+3y^{(3)}+3\ddot{y}+4\dot{y}+5y=\mathrm{e}^{-3t}+\mathrm{e}^{-5t}\sin(4t+\pi/3)$,且方程的初值为 $y(0)=1,\dot{y}(0)=\ddot{y}(0)=1/2,y^{(3)}(0)=0.2$,试用Simulink搭建系统的仿真模型,并绘制出仿真结果曲线。

5.5 考虑习题5.4的模型,假设给定的微分方程变化成时变线性微分方程:

$$y^{(4)}+3ty^{(3)}+3t^2\ddot{y}+4\dot{y}+5y=\mathrm{e}^{-3t}+\mathrm{e}^{-5t}\sin(4t+\pi/3)$$

而方程的初值仍为 $y(0)=1,\dot{y}(0)=\ddot{y}(0)=1/2,y^{(3)}(0)=0.2$,试用Simulink搭建起系统的仿真模型,并绘制出仿真结果曲线。

5.6 试将下面两个多变量传递函数矩阵[11]用Simulink表示出来,并绘制出单位负反馈系统的阶跃

响应曲线。

① $\boldsymbol{G}_1(s)=\begin{bmatrix} \dfrac{0.5\mathrm{e}^{-0.2s}}{3s+1} & \dfrac{0.07\mathrm{e}^{-0.3s}}{2.5s+1} & \dfrac{0.04\mathrm{e}^{-0.03s}}{2.8s+1} \\ \dfrac{0.004\mathrm{e}^{-0.5s}}{1.5s+1} & \dfrac{-0.003\mathrm{e}^{-0.2s}}{s+1} & \dfrac{-0.001\mathrm{e}^{-0.4s}}{1.6s+1} \end{bmatrix}$

② $\boldsymbol{G}_2(s)=\begin{bmatrix} \dfrac{0.66\mathrm{e}^{-2.6s}}{6.7s+1} & \dfrac{-0.0049\mathrm{e}^{-s}}{9.06s+1} \\ \dfrac{1.11\mathrm{e}^{-6.5s}}{3.25s+1} & \dfrac{-0.012\mathrm{e}^{-1.2s}}{7.09s+1} \\ \dfrac{-33.68\mathrm{e}^{-9.2s}}{8.15s+1} & \dfrac{0.87(11.61s+1)\mathrm{e}^{-s}}{(3.89s+1)(18.8s+1)} \end{bmatrix}$

5.7 已知微分方程可以表示为[12]：

$$\begin{cases} \dot{u}_1=u_3 \\ \dot{u}_2=u_4 \\ 2\dot{u}_3+\cos(u_1-u_2)\dot{u}_4=-\mathrm{g}\sin u_1-\sin(u_1-u_2)u_4^2 \\ \cos(u_1-u_2)\dot{u}_3+\dot{u}_4=-\mathrm{g}\sin u_2+\sin(u_1-u_2)u_3^2 \end{cases}$$

其中，$u_1(0)=45, u_2(0)=30, u_3(0)=u_4(0)=0$，g= 9.81，试用Simulink求解此微分方程，并绘制出各个状态变量的时间曲线。

5.8 已知Apollo卫星的运动轨迹(x,y)满足下面的方程：

$$\begin{cases} \ddot{x}=2\dot{y}+x-\dfrac{\mu^*(x+\mu)}{r_1^3}-\dfrac{\mu(x-\mu^*)}{r_2^3} \\ \ddot{y}=-2\dot{x}+y-\dfrac{\mu^* y}{r_1^3}-\dfrac{\mu y}{r_2^3} \end{cases}$$

其中$\mu=1/82.45, \mu^*=1-\mu, r_1=\sqrt{(x+\mu)^2+y^2}, r_2=\sqrt{(x-\mu^*)^2+y^2}$，假设系统初值为$x(0)=1.2, \dot{x}(0)=0, y(0)=0, \dot{y}(0)=-1.04935751$，试搭建起Simulink仿真框图并进行仿真，绘制出Apollo位置的(x,y)轨迹。

5.9 试求出隐式微分方程：

$$\begin{cases} \dot{x}_1\ddot{x}_2\sin(x_1x_2)+5\ddot{x}_1\dot{x}_2\cos(x_1^2)+t^2x_1x_2^2=\mathrm{e}^{-x_2^2} \\ \ddot{x}_1x_2+\ddot{x}_2\dot{x}_1\sin(x_1^2)+\cos(\ddot{x}_2x_2)=\sin t \end{cases}$$

的数值解，$x_1(0)=1, \dot{x}_1(0)=1, x_2(0)=2, \dot{x}_2(0)=2$，并绘制出轨迹曲线。

5.10 考虑延迟微分方程：

$$y^{(4)}(t)+4y^{(3)}(t-0.2)+6\ddot{y}(t-0.1)+6\ddot{y}(t)+4\dot{y}(t-0.2)+y(t-0.5)=\mathrm{e}^{-t^2}$$

且在$t\leqslant 0$时该方程具有零初始条件，试用Simulink建模的方式直接求解该微分方程，并绘制出$y(t)$曲线。

5.11 考虑下面给出的延迟微分方程模型：

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t}=\frac{0.2y(t-30)}{1+y^{10}(t-30)}-0.1y(t)$$

假设$y(0)=0.1$，试用Simulink搭建仿真模型，并对该系统进行仿真，绘制出$y(t)$曲线。

5.12 假设已知分数阶线性微分方程为[13]：

$$0.8\mathscr{D}_t^{2.2}y(t)+0.5\mathscr{D}_t^{0.9}y(t)+y(t)=1,\ y(0)=\dot{y}(0)=\ddot{y}(0)=0$$

试求该微分方程的数值解。若将微分阶次2.2近似成2，0.9阶近似成1阶，则可以将该微分方程近似为整数阶微分方程，试比较整数阶近似的计算精度。

5.13 试用近似方法求解下面的分数阶非线性微分方程，假设初始条件都是0，求解方程并验证得出解的正确性。

$$\frac{3\mathscr{D}^{0.9}y(t)}{3+0.2\mathscr{D}^{0.8}y(t)+0.9\mathscr{D}^{0.2}y(t)}+\left|2\mathscr{D}^{0.7}y(t)\right|^{1.5}+\frac{4}{3}y(t)=5\sin(10t)$$

5.14 设分数阶非线性微分方程由图5-100中的Simulink模型描述，试写出该微分方程的数学表达式，并绘制出输出信号$y(t)$。

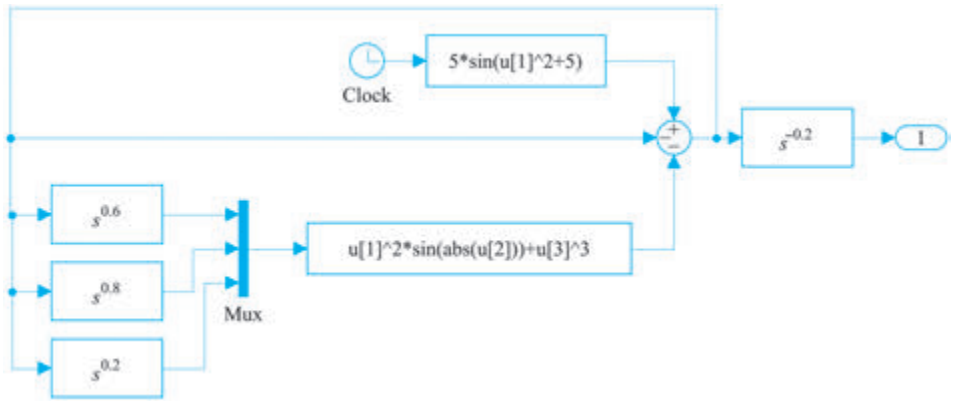


图5-100 非线性分数阶微分方程的Simulink描述（文件名：c5mfode4.slx）

5.15 考虑大时间延迟受控对象模型$G(s)=\dfrac{10\mathrm{e}^{-20s}}{2s+1}$，假设控制器模型为$G_{\mathrm{c}}(s)=0.0249+\dfrac{0.00342}{s}$，试同时用曲线和表盘的形式显示控制效果。

5.16 考虑如下Lorenz方程模型，该模型没有输入信号：

$$\begin{cases}\dot{x}_1(t)=-\beta x_1(t)+x_2(t)x_3(t)\\ \dot{x}_2(t)=-\rho x_2(t)+\rho x_3(t)\\ \dot{x}_3(t)=-x_1(t)x_2(t)+\sigma x_2(t)-x_3(t)\end{cases}$$

假设选择其3个状态变量$\boldsymbol{x}_i(t)$为其输出信号，以β、σ、ρ和$\boldsymbol{x}_i(0)$向量为附加参数，试将该模块封装起来，并绘制在不同参数下的Lorenz方程解的三维曲线。

5.17 假设已知误差信号$e(t)$，试构造出求取ITAE、ISE、ISTE准则的封装模块。要求：误差信号$e(t)$为该模块的输入信号，双击该模块弹出一个对话框，允许用户用列表框的方式选择输出信号形式，将选定的ITAE、ISE、ISTE之一作为模块的输出端显示出来。其中ISTE的定义为：

$$J_{\mathrm{ISTE}}=\int_0^t\tau e^2(\tau)\mathrm{d}\tau$$

5.18 试将van der Pol方程：

$$\begin{cases}\dot{x}_1(t)=x_2(t)\\ \dot{x}_2(t)=-\mu(x_1^2(t)-1)x_2(t)-x_1(t)\end{cases}$$

对应的Simulink模型封装起来，使得封装参数为参数μ和两个积分器的初值x_{10}、x_{20}，封装模块没有输入端口，由两路输出端口，返回$x_1(t)$和$x_2(t)$。如果让参数μ为输入端口，试重新建立Simulink模型并重新封装该模块。

参考文献

[1] 薛定宇. 薛定宇教授大讲堂（卷VI）：Simulink建模与仿真 [M]. 北京：清华大学出版社，2021.

[2] Atherton D P. Nonlinear control engineering:describing function analysis and design [M]. London: Van Nostrand Reinhold,1975.

[3] 薛定宇. 高等应用数学问题的MATLAB求解 [M]. 5版. 北京:清华大学出版社,2023.

[4] Liberzon D, Morse A S. Basic problems in stability and design of switched systems[J]. IEEE Control Systems Magazine,1999,19(5):59–70.

[5] Oustaloup A, Levron F, Nanot F, et al. Frequency band complex non integer differentiator: characterization and synthesis[J]. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,2000,47(1):25–40.

[6] 薛定宇. 控制系统计算机辅助设计:MATLAB语言与应用 [M]. 5版. 北京:清华大学出版社,2026.

[7] 薛定宇,白鹭. 分数阶微积分学:数值算法与实现 [M]. 北京: 清华大学出版社,2023.

[8] 汪成为,高文,王行仁. 灵境(虚拟现实)技术的理论、实现与应用 [M]. 北京:清华大学出版社,1996.

[9] 严子翔. VRML虚拟现实网页语言. 北京:清华大学出版社,2001.

[10] Frederick D K,Rimer M. Benchmark problem for CACSD packages[C]// Chen Z Y and Liu C C eds. Computer Aided Design in Control Systems,Selected Papers from the 4th IFAC Symposium. Beijing:Pergamon Press,1988.

[11] 李少远,蔡文剑. 工业过程辨识与控制 [M]. 北京:化学工业出版社,2005.

[12] Molor C B. Numerical computing with MATLAB[Z]. MathWorks Inc,2004.

[13] Podlubny I. Fractional differential equations[M]. San Diego:Academic Press,1999.